VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Filip Kusy

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

POKROČILÉ METODY ZABEZPEČENÍ SÍTĚ PROTI ÚTOKŮM
ADVANCED NETWORK SECURITY METHODS AGAINST ATTACKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Filip Kusy

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Ondřej Krajsa, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Filip Kusy ***ID:*** 154785
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Pokročilé metody zabezpečení sítě proti útokům

POKYNY PRO VYPRACOVÁNÍ:

Analyzujte možnosti zabezpečení sítí využívajících prvky Mikrotik a servery s OS Linux. Na základě analýzy navrhněte a realizujte zabezpečení sítě s prvky pro detekci útoků a jejich analýzu.

DOPORUČENÁ LITERATURA:

[1] Chapter 9 - Network forensics. The Basics of Digital Forensics. 2015, s. 133-144. DOI: 10.1016/B978-0--2-801635-0.00009-7. ISBN 978-0-12-801635-0.

[2] Using Snort for Network-Based Forensics-Chapter 5. Digital Forensics for Network, Internet, and Cloud Computing. 2010, s. 113-132. DOI: 10.1016/B978-1-59749-537-0.00005-3. ISBN 978-1-59749-537-0.

[3] BURGESS, Dennis. Learn RouterOS. Lexington]: Dennis Burgess, 2009, 391 s. : il. ISBN 978-0-557-09271-0.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto študentská práca sa venuje oblasti zabezpečenia proti sieťovým útokom. Zoberá sa problematike útokov a možnostiam ako im zabrániť. Následne sa zaoberá skúmaním Snort Varianty IPS/IDS sytému. Zaoberá sa prepojením medzi Mikrotikom a Linuxovým serverom s programom Snort.

KĽÚČOVÉ SLOVÁ

Snor, Mikrotik, Firewall, IDS, IPS, Surikata, Útoky, Linux, Server, PHP, PulledPork

ABSTRACT

This student work focuses on security against network attacks. It focus on network attacks and ways to prevent them. Subsequently, it deals with the Snort variant of the IPS/IDS system. It deal with the connection between Mikrotik and the Snort Linux server.

KEYWORDS

Snor, Mikrotik, Firewall, IDS, IPS, Surikata, Attacks, Linux, Server, PHP, PulledPork

KUSY, Filip *Pokročilé metódy zabezpečenia siete proti útokom*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2018. 67 s. Vedúci práce bol Ing. Ondřej Krajsa, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Pokročilé metódy zabezpečenia siete proti útokom“ vypracoval(a) samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Rád by som Poďakoval vedúcemu diplomovej práce pánovi Ing. Onřejovi Krajsovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto diplomovej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Náplňou diplomovej práce je analyzovanie možností zabezpečenia sietí využívaných prvky Mikrotik a serverov s OS Linux. Na základe analýzy navrhnúť a realizovať zabezpečenie siete s prvkami pre detekciu útokov a ich analýzu.

Prvá časť práce sa venuje firewalom a prechádza až k skúmaniu IDS a IPS systémov. Nadbieha k detailom a princípom oboch druhov systémov. Následne sa venuje konkrétnym najznámejším predstaviteľom IDS/IPS systémov. Vyberá Hlavného predstaviteľa týchto systémov a bližšie špecifikuje jeho vlastnosti.

V ďalšej časti sa táto práca zaoberá typmi útokov a ich detekcii. Potom v praktickej časti prichádza na rad samotná realizácia. Ukazuje sa tu základná konfigurácia Mikrotiku, Linuxového servera a následne programu Snort. Popisujeme tu nastavovanie nástroja Mikrotiku na preposielanie paketov na konkrétnu IP adresu, konfiguráciu Linuxovej distribúcie s jednotlivými balíčkami, pričom zahrnuté sú hlavne príkazy na nastavovanie a úpravu jednotlivých súborov, či nastavení.

V poslednej časti práce sú prezentované realizácia a testy IPS systému s programom snort a pridruženými pravidlami od cisco nástroja PulledPork. Práca teda vysvetľuje akými spôsobmi je možné pracovať s detekovanými útokmi a zabraňovaniu nim. Práca sa opiera o reálne namerané a skúmané útoky na systém. Taktiež sú definované a priblížené úpravy systému na zrýchlenie zabezpečenia proti útokom na Linuxový server.

1 TEORETICKÁ ČASŤ

1.1 Firewally

1.1.1 Čo je to firewall

Firewall je sieťové bezpečnostné zariadenie na monitorovanie prichádzajúcej a odchádzajúcej komunikácie a rozhodovanie o povolení, či blokovaní špecifického prenosu založeného na definovaných bezpečnostných pravidlách. Kontrola údajov prebieha na základe aplikovania pravidiel, ktoré určujú podmienky a akcie. Podmienky sa stanovujú pre údaje, ktoré možno získať z dátového toku (napr. zdrojová, cieľová adresa, zdrojový alebo cieľový port a rôzne iné). Úlohou firewallu je vyhodnotiť podmienky, pri ktorých splnení je vykonávaná akcia. Dve základné akcie sú „povoliť dátový tok“ a „zamietnuť dátový tok“.

Brány firewall boli prvou líniou obrany v oblasti bezpečnosti siete viac ako 25 rokov. Vytvárajú bariéru medzi zabezpečenými a kontrolovanými internými sieťami, ktoré môžu byť dôveryhodné a nedôveryhodné mimo siete, ako napríklad internet. Pričom firewall môže byť riešený hardvérovo alebo softwarovo.

Firewall môže byť teda realizovaný ako spustený program v počítači, alebo ako samostatné hardvérové zariadenie. Programové riešenie však spravidla ponúka vyšší komfort údržby a je možné ho obsluhovať aj s pomerne malými znalosťami problematiky. Daň za tento komfort je však pomalejšie vybavovanie požiadaviek. Na druhú stranu, hardvérový firewall (zariadenie) ponúka maximálnu priepustnosť požiadavkov za cenu nižšieho komfortu pri správe a údržbe. K obsluhe hardvérového firewallu sú spravidla nutné hlbšie znalosti. [1]

1.1.2 Delenie firewallov

Prehľad firewallov. Firewally možno deliť podľa najvyššej vrstvy, z ktorej firewall zbiera informácie k rozhodnutiu o filtrovaní komunikácie. Zaujímavé sú pre nás tieto vrstvy:

- **Sieťová vrstva** - najrýchlejšia a spravidla najmenej nákladná varianta firewallu. Filtruje len veľmi na hrubo, len na základe zdrojovej, alebo cieľovej IP adresy obsiahnutej v záhlaví analyzovaného paketu. Je teda možné zakázať napríklad prevádzku prichádzajúcu z nežiadúcich sietí alebo do týchto sietí naopak smerujúcu. V dnešnej dobe distribuovaných útokov sa jedná o prakticky nepoužiteľnú variantu, preto sa čisto v tejto podobe používa len v špeciálnych prípadoch.

- **Transportná vrstva**
  - **Bezstavový** - okrem údajov zo sieťovej vrstvy (IP adresy) berie v úvahu aj hodnoty transportných portov. Súčasná generácia týchto filtrov je veľmi rýchla a bežne býva označovaná ako paketové firewally, napriek tomu, že filtruje de facto na úrovni segmentov. Pre ich veľkú rýchlosť je vhodné ich nastaviť na miesta s hustou prevádzkou, napríklad vstupná brána do siete a podobne. Nastavenie takéhoto typu firewallu vyžaduje pomerne dobré znalosti sieťových technológií.
  - **Stavový** - tento firewall oproti bezstavovému dokáže rozlíšiť už naviazané spojenia od ostatnej prevádzky a nemu pridruženú komunikáciu ľahšie prepustí. Funguje to tak, že každý paket je najskôr posudzovaný vzhľadom k zoznamu existujúcich (už povolených) spojení. Ak k niektorému patrí, je prepustený bez ďalšej analýzy. Ak nie, prechádza štandardným paketovým filtrom a môže byť prepustený alebo zahodený. Nasadenie stavového firewallu je dôležité, v prípade, že chceme efektívne a správne filtrovať komunikáciu postavenú na transportnom protokole TCP. Teda u väčšiny internetových protokolov. Daň za kvalitnejšie filtrovanie je zrejmá. Nižšia rýchlosť filtrovania, než u paketových filtrov, ktorá súvisí s vyššou náročnosťou tohto typu firewallu, pričom je veľmi závislá na počte aktívnych spojení.
- **Aplikačná vrstva** - najvyššia vrstva, na ktorej môže firewall pracovať je aplikačná. Tieto firewally spravidla naplno „rozumejú“ fungovaniu aplikácií a protokolov a sú schopné detektovať napr. prípady ich neštandardného chovania apod. Filtrujú veľmi jemne, za cenu nižšej rýchlosti. Najvhodnejšie je ich použiť až na koncové stanice. Firewall pracujúci na aplikačnej vrstve môže byť tiež označovaný ako proxy firewall, čo súvisí s tým, že jeho úloha je obdobná ako u proxy serverov - vytvoriť prostredníka v komunikácii. V oblasti firewallov sa však nejedná len o zaistenie anonymity, ale tiež o komplexnú analýzu a filtrovanie komunikácie. Konfigurácia aplikačného firewallu môže byť veľmi jednoduchá alebo naopak veľmi zložitá. Záleží na tom, do akej hĺbky nadstavení je jeho užívateľ schopný zájsť.

Pri filtrácii komunikácie je dôležité vykonávať záznam (log) o predchádzajúcich paketoch, často len o neprepustených paketoch. Analýzou týchto záznamov je možné donadstaviť politiku firewallu a dosiahnuť tak čo najlepšieho filtrovania paketov. Niekedy môže byť užitočné, keď firewall loguje aj správy, ktoré prepustil. Je to možné využiť napr. ku statickým účelom alebo k upraveniu nadstavení pravidiel firewallu, v prípade, že prepustí prevádzku, ktorú by mal blokovať. V prípade, že dôjde na firewalle k zahodeniu paketov. je možné (ale nie nutné) o tejto skutočnosti informovať zdrojovú stanicu prostredníctvom chybového hlásenia. Na tento účel sa

používa protokol ICMP, konkrétne správa „nedoručený IP datagram“(destination unreachable). Zrejmou nevýhodou zasielania tejto odpovede je, že sa útočník dozvie o existencii obranného systému a že získa potencionálnu možnosť, ako zariadenie zahltiť. Z tohoto dôvodu sa možnosti informovania zdroja o zahodení paketu príliš nevyužíva.

Pri filtrovaní komunikácie je potrebné myslieť na obidva smery komunikácie. Niekedy je dostatočné, ak sa zabezpečí jen jeden(prichádzajúci) smer. Tento spôsob je aplikovaný u základných typov firewalov. Ak budeme prevádzkovať napríklad webový server, základným cieľom je ochrániť ho pred nechcenou komunikáciou z vonku. Takýto webový server by nemal generovať žiadne odchádzajúce spojenie, len očakáva požiadavky pochádzajúce od klientov. Ak sa však nejaké spojenie generovať bude, je pravdepodobné, že niečo nie je v poriadku a je teda dobré mať nakonfigurovaný firewall aj v odchádzajúcom smere, ktorý tejto komunikácii buď zabráni alebo ju aspoň zapíše do záznamu(logu). [2] [3]

1.1.3 Pravidlá firewallov

Paketové firewally sú konfigurované pomocou pravidiel. Tieto pravidlá sa musia radiť do postupnosti, kvôli náväznosti a postupu ich spracovávania. Prichádzajúci paket sa kontroluje na všetky pravidlá postupne podľa poradia pravidiel, pričom je prepustený alebo naopak zahodený iba ak sa nájde vyhovujúce pravidlo. Ak sa ale skontroluje podľa všetkých pravidiel a ani jedno nevyhovuje, použije sa takzvané východzie(implicitné) pravidlo. Implicitné pravidlo môže teda paket prepustiť, alebo naopak zahodiť. Najčastejšie sa však toto implicitné pravidlo spája so zahadzovaním. Predchádzajúce pravidlá teda pakety(služby) povoľujú a všetko ostatné je zakázané. [3]

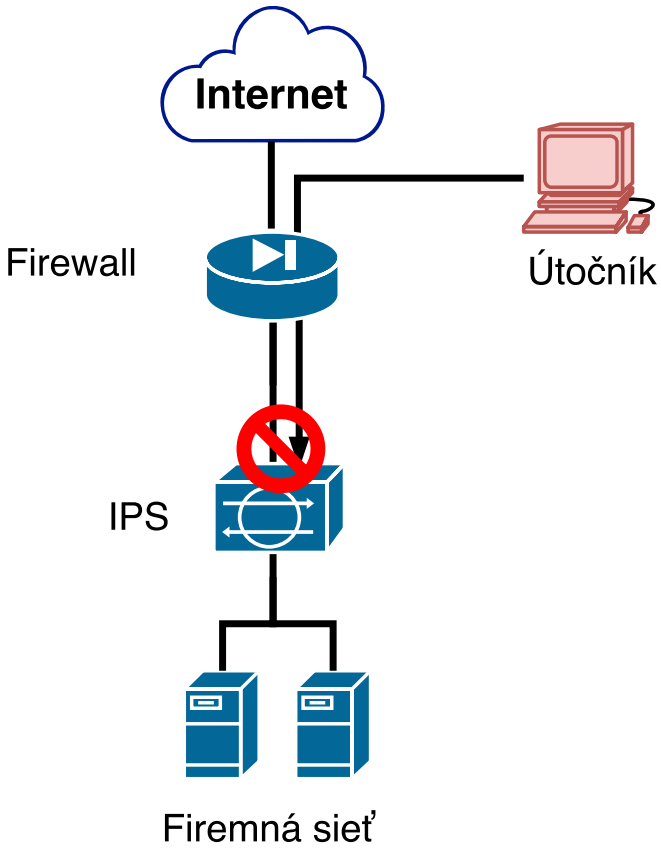
Tab. 1.1: Príklad konfigurácie paketového firewallu

Smer Direction	Protokol Protocol	Zdrojová IP adresa Source address	Zdrojový port Source port	Cieľová IP adresa Destination address	Cieľový port Destination port	Akcia Action
odchádzajúci (outgoing)	IP	10.10.1.0	ľubovolný (any)	ľubovolná (any)	ľubovolný (any)	povoliť (allow)
prichadzajúci (comming)	TCP	ľubovolná (any)	ľubovolný (any)	10.10.1.5	80	povoliť (allow)
obojsmerný(outgoing/comming)	IP	ľubovolná (any)	ľubovolný (any)	ľubovolná (any)	ľubovolný (any)	zakázať (deny)

1.2 IPS/IDS systémy

1.2.1 IPS - Intrusion Prevention System (IPS)

Môže byť riešené aktívnym fyzickým zariadením, alebo softwatovo. IPS zariadenie je medzi výstupom jedného interface a vstupom druhého interface. Tento systém testuje aktuálne dátové pakety, ktoré prechádzajú sieťou. Pracuje v reálnom čase a umožňuje paketom prejsť, alebo naopak zabraňuje prieniku do siete. [4]



Obr. 1.1: IPS

Ako IPS pracuje

IPS býva často krát zariadenie, ktoré je v sieťovej infraštruktúre zaradené hneď za firewallom a poskytuje tak doplnkový stupeň ochrany pred vstupom škodlivých dát a komunikácie do systému. Systém IPS je radený priamo medzi odosielateľa a prijímateľa sieťovej komunikácie a dokáže tak priamo, aktívne a efektívne skenovať takúto komunikáciu. Pokiaľ sa vyskytne nejaká anomália, ktorá môže byť vyhodnotená ako narušenie bezpečnosti, IPS začne konať. Medzi činnosti, ktoré IPS vykonáva pri odhalení škodlivej komunikácii patrí :

- Upozornenie správcu na výskyt bezpečnostného problému
- Zahadzovanie škodlivých, resp. podozrivých paketov
- Blokovanie kompletnej komunikácie z konkrétnej IP adresy
- Obnovenie alebo resetovanie spojenia

IPS zariadenie radené priamo do sieťovej komunikácie svoju činnosť vykonáva v reálnom čase, preto musí pracovať rýchlo a efektívne, aby zbytočne nezaťažovalo sieť a neznižovalo tak jej výkon. IPS musí byť tiež schopné rozlíšiť, či sa skutočne jedná o škodlivú komunikáciu, aby sa predišlo problémom pri komunikácii spôsobenými práve činnosťami ktoré vykonáva IPS pri detekcii nechcenej komunikácie. [4] [8]

Detekčné mechanizmy

Pri IPS systémoch sa využíva veľké množstvo detekčných mechanizmov na nájdenie **exploitov**, avšak si spomenieme dva dominantné, ktoré sú využívané najčastejšie a sú najrozšírenejšie.

A. Porovnávanie vzoriek (Signature-based detection) Detekčný mechanizmus je založený na vyhľadávaní fixných sekvencií znakov v pakete ktoré sú typické pre niektoré druhy sieťových útokov. Ako už samotný názov detekčného mechanizmu naznačuje, jedná sa o značne nepružnú, avšak jednoduchým spôsobom použiteľnú metódu. Systém IPS má v tomto prípade svoj vlastný slovník „značiek“, ktorý si postupne automaticky dopĺňa. IPS monitoruje sieťový tok a pokiaľ nájde v komunikácii značku, ktorá je zhodná s tou v slovníku, vykoná určitú príslušnú operáciu

V mnohých prípadoch je vzorka porovnateľná len vtedy, ak je podozrivý paket priradený konkrétnej službe, prípadne (pri požiadavke na vyššiu presnosť detekcie) je paket cielený na konkrétny port, alebo prichádzajúci z konkrétneho zdrojového portu. To umožňuje znížiť počet vykonaných kontrol nad každým paketom, ale zvyšuje to náročnosť detekcie pri protokoloch, ktoré sa neodkazujú na známe porty (napr. Trójske kone a nimi vyvolané toky údajov, ktoré si aplikácie zvyčajne sami presúvajú podľa potreby).

Štruktúra mechanizmu porovnávania vzoriek je približne nasledovná:

Ak je TCP paket typu IPv4, cieľový port je 2222 a údaje obsahujú reťazec znakov “pqr” spustí sa reakcia.

Tento príklad detekčného mechanizmu založeného na porovnávaní vzoriek je samozrejme veľmi zjednodušený. V skutočnosti sa do vzorky vkladá i špecifický štartovací a ukončovací bod pre detekciu v rámci paketu, prípadne sú špecifikované TCP príznaky pre pakety na ktoré má byť braný ohľad.

Výhody:

- Táto technika je veľmi jednoduchou, až primitívnou cestou pre detekciu prienikov.

- Umožňuje priamu koreláciu exploitov podľa veľmi špecifickej vzorky.
- Spoľahlivejšie hlási presne špecifikovanú vzorku.
- Je aplikovateľná naprieč všetkými protokolmi.

Nevýhody:

- Táto metóda môže viesť k vyššiemu množstvu falošných hlásení, ktoré nie sú totožné s hláseniami, ktoré boli očakávané v opise signatúr.
- Akékoľvek modifikácie útoku môžu viesť k opomenutiu detekcie incidentu (false negative).
- Môže vyžadovať niekoľko rôznych signatúr pre obsluhu toho istého typu ohrozenia. Rôzne nástroje útoku generujú potrebu mnohorakých signatúr.
- Je zvyčajne limitovaná na detekciu jednoduchého paketu a z toho dôvodu nepodporuje dostatočným spôsobom detekciu incidentov v prúde údajov (napr. v protokole HTTP). Tento scenár nepriamo podporuje jednoduché použitie tzv. „vyhýbacích“ (evasion) techník útokov.

B. Analýza odchýlok (statistical anomaly-based detection) Metóda štatistickej analýzy odchýlok je založená na vyhľadávaní takej sieťovej prevádzky, ktorá je odlišná od prevádzky označenej za „normálnu“. Pri tomto mechanizme detekcii, je najprv vytvorený obraz normálneho prevádzkového stavu siete a ten je potom periodicky porovnávaný s aktuálnym stavom siete. IPS takto dokáže detekovať anomálie v sieťovej komunikácii a vykonať príslušné opatrenia pre ochránenie systému. Najväčším problémom tejto metodiky je prvotná definícia „normálnej“ prevádzky.

Niektoré systémy majú pevne stanovené definície normálu a na základe toho môžu byť hodnotené heuristickými metódami. Niektoré systémy sú vybudované tak, aby sa po implementácii postupne dostali do normálu, pričom ich výhodou je, že sú schopné eliminovať možnosť nevhodnej klasifikácie abnormálneho správania za normálne.

Ak je vzorka sledovanej prevádzky považovaná za normálnu, systém musí rozhodnúť, akým spôsobom rozlíšiť medzi povolenými odchýlkami a odchýlkami ktoré nie sú povolené, alebo ktoré môžu predstavovať prebiehajúci útok. Aplikácie v tejto oblasti sú prevažne akademické, i keď existuje aj niekoľko komerčných produktov, ktoré používajú metódu analýzy odchýlok.

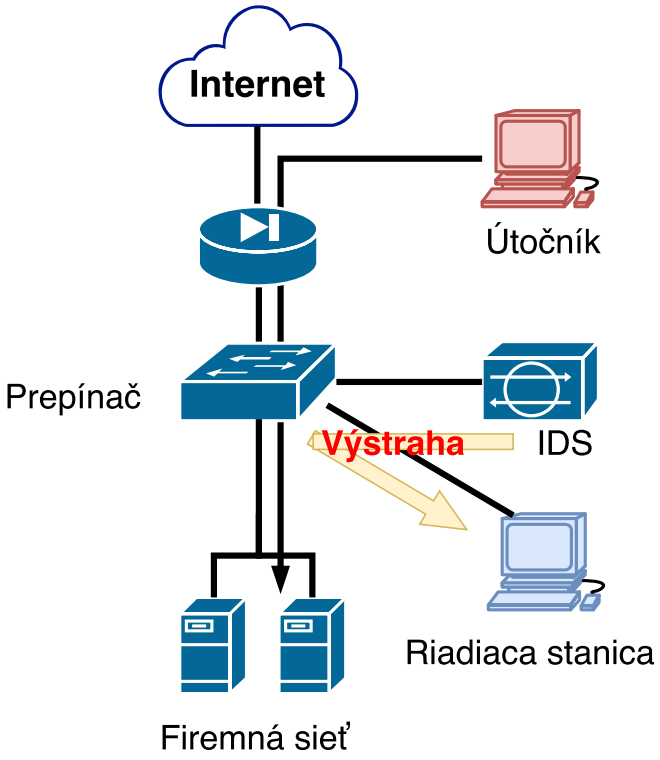
Podkategóriou tohoto typu detekcie sú metódy založené na detekcii profilov (profile-based analysis). Tieto systémy majú hlásenia založené na zmenách spôsobu akým prebieha interakcia používateľov a siete. Druhy tejto interakcie spôsobujú mnohé podobné limitácie a na základe týchto zmien je možné odvodiť a definovať jednotlivé profily.

Zaujímavé fakty môžu byť odvodené z trendových údajov, pričom tieto sú schopné odhaliť prebiehajúci útok na základe uvedeného algoritmu.

Informácie, ktoré tieto systémy poskytujú sú však vo všeobecnosti veľmi nešpecifické a ak majú byť uvedené do správneho kontextu, je nutné vždy ich ďalšie došetrenie.
V niektorých prípadoch sú hranice medzi jednotlivými metodikami rozostreté, pretože mnohé prostriedky pre dekódovanie protokolov upozorňujú používateľa na prítomnosť porušenia pravidiel protokolu, ktoré nemusia priamo súvisieť s konkrétnym typom útoku, avšak sa zdajú byť „neobvyklé“ (napr. detekcia možného pretečenia zásobníka). V takomto prípade detekčné prostriedky udržujú databázu atribútov systému pre detekciu anomálií. [6] [7]

1.2.2 IDS - Intrusion Detection System (IDS)

Je softwarovo alebo hardwarovo založené riešenie, sledujúce sieťové prenosy pasívnym spôsobom. Dátový prenos nejde priamo cez IDS zariadenie, ale namiesto toho si IDS zariadenie monitoruje prenos prechádzajúci cez interface sieťového zariadenia. Ak dôjde k detekcii škodlivého prenosu, IDS pošle upozornenie danej riadiacej stanici.[5]



Obr. 1.2: IDS

Ako IDS pracuje

Systém IDS potrebuje iba detekovať hrozby a ako taký je umiestnený mimo pásma sieťovej infraštruktúry, čo znamená že sa nenachádza v skutočnej komunikačnej ceste medzi odosielateľom a príjemcom informácií. V IDS riešenia bývajú častokrát použité výhody TAP alebo SPAN portu na analyzovanie kópie prevádzky (toto zabezpečuje nezníženie sieťového výkonu pri používaní IDS systému). Hlavné spôsoby monitorovania útokov sú založené na detekcii určitej známej sekvencii znakov v komunikácii, detekovaní neobvyklej komunikácie a analýze protokolov. Ide v podstate o program , ktorý sa stará o identifikáciu prienikov, hardvér, na ktorom beží súbor pravidiel(bezpečnostná politika), ktoré vyvolajú príslušné činnosti pri zistení prieniku. Systémy IDS nedetekujú vždy len známe útoky, ale snažia sa zachytiť aj tzv.: „pre-attack probes“, čo sú akcie útočníka pred zahájením útoku. Na základe získaných informácií z pozorovanej komunikácie, môžu systémy detekcie vniknutia poskytovať tieto informácie administrátorovi, v reálnom čase zmeniť časť konfigurácie systému a tak zamedziť potenciálnemu prieniku. Prípadne tieto systémy môžu sledovať a kontrolovať ďalšie útočníkove kroky a aktívne tak zamedziť deštruktívnym následkom. Niektoré z IDS systémov môžu byť konfigurované na zachytávanie paketov súvisiacich s podozrivým obsahom. Zaznamenávanie je možné nadstaviť od zaznamenávania len podozrivých paketov, ktoré určuje IDS, až po zachytávania celých úsekov komunikácie. Zaznamenávanie celých častí komunikačných prenosov je možné nadstaviť na niektorých IDS za účelom spätného dokladania konkrétneho komunikačného vlákna. Treba si však uvedomiť, že hlavnou úlohou IDS systémov je detekcia útoku s následným oboznámením administrátora. IDS teda neslúži na zabránenie útoku, je len akýmsi pomocníkom.

Špeciálnym prípadom IDS systému, ktorý v sebe zahŕňa aj protiopatrenia voči útokom sa nazýva IDPS(Intrusion Detection and Prevention System). Protiopatrenia však nie sú definované samostatne, ale sú to reakcie na detekčnú časť IDPS systému. [5][8]

Deteckiu prienikov rozdeľujeme podľa princípu podobne ako u IPS systémov na :

A. Detekcie štatistickej odchýlky Pracujúcej na princípe vyhľadávania štatisticky významnej odchýlky voči stanovenému normálu. Stavový normál je pritom udávaný normálovými množinami ktoré sú definované pre subjekty a objekty(nie len užívatelia, ale aj samotné pracovné stanice, servery, súbory, sieťové body atď.). Na základe empirických znalostí(získavaných zo skúseností) sa pre všetky skupiny určia normálové hodnoty a ich tolerancia.

B. Porovnávanie vzorov Pracujúcej na princípe porovnávania komunikácie (komunikačného reťazca) so vzorom. Vzory môžu zodpovedať jednoduchým udalostiam alebo sekvenciám udalostí. Je to vlastne určitý druh filtrácie veľkého objemu dát. [9]

Koponenty IDS systémov: Senzory, Agenti, Manažér

Senzory

Sú zodpovedné za vstup do prebiehajúceho spojenia sledovanie jeho aktivít. Taktiež majú za úlohu zbierať dáta o prevádzke a odovzdávanie ich agentom. Ich umiestnenie je veľmi dôležité. Pri umiestnení za firewallom, zachytávajú len komunikáciu, ktorá je od firewallu povolená. Takto môže nastať prípad, že senzory nezaznamenajú udalosť napadnutia. Na druhú stranu, pri umiestnení senzorov pred firewall, môže spôsobiť veľa poplašných správ, ktoré nemusia byť nutne hrozbou.

Agenti

Analyzujú v jednotlivých uzloch siete zozbierané dáta. Pri zachytení podozrivej aktivity posielajú správu manažérovi. Pri niektorých vylepšených IDS systémoch sú používaní vzájomne distribuovaní agenti, ako technika na zvýšenie okamžitej odpovede na útok a na výrazné zdokonalenie detekcie útokov napríklad typu DoS.

Manažér

Vykonáva danú operáciu podľa konfigurácie administrátora na základe prichádzajúcej správy. IDS poskytuje viacero rôznych operácií, ktoré môže manažér vykonať na základe svojej štruktúry. Je dokonca možné, že IDS môže mať viac ako jedného manažéra. U takýchto systémoch si manažéri medzi sebou vymieňajú informácie o narušení siete vzájomnou komunikáciou. [5]

Delenie IDS podľa prístupu ochrany:

- **Uzlovo orientované IDS (HOST-Based IDS) HIDS**
  Boli vyvíjané a implementované ako prvé. Vďaka tomu, že sú inštalované na lokálnych zariadeniach sú všestrannými v porovnaní so sieťovo orientovanými IDS(NIDS). Môžu byť inštalované na rôzne druhy serverov, pracovných staniciach alebo notebookoch. HIDS analyzuje a zbiera dáta prichádzajúce do lokálneho zariadenia. Kontrola je zabezpečovaná v systémových logovacích súboroch. Zozbierané dáta sa potom analyzujú buď na lokálnom zariadení , alebo sa posielajú do centralizačného zariadenia. HIDS sú účinné a využívané v detekovaní vnútorných pokusoch o útok. Treba si však uvedomiť, že pri rozsiahlej sieti a veľmi veľkom počte koncových zariadení môže IDS spôsobovať zníženie rýchlosti siete a neefektívnosť siete. Navyše, ak sa útočníkovi podarí zablokovať zozbierané dáta koncového bodu, nebude IDS účinné. [10]

Príklady implementácie HIDS: Win NT/2000 security Event Logs, Enterprise Management systems audit data, RDMS audit sources

- **Sieťovo orientované IDS(Network-Based IDS) NIDS**
  Najviac využívané v sietiach so prepínačmi(Switchmi). Kvôli tomu, že pri prepínačoch prúdia pakety vždy presne daným smerom. U tohoto druhu systémov je však nutné správne určenie umiestnenia senzorov, aby bol zabezpečený odchyt čo najväčšieho množstva komunikácie. NIDS používajú na zachytávanie prevádzky nástroje ako napríklad „packet-sniffer“ pracujúcej na TCP/IP alebo iných sieťových protokoloch. Pakety sú kontrolované a porovnávané, či nie sú útokmi. Vo všeobecnosti sa dá povedať, že NIDS je najlepší v detekovaní vonkajších neautorizovaných prístupoch a útokoch typu DoS. V prípade že je komunikácia kryptovaná alebo ide o vysoko rýchlostnú sieť, vyskytujú sa u NIDS problémy s odchytom paketov a ich prekladom. [10]
  Príklady implementácie NIDS: Snort, Dragon, Shadow, NFR, NetProwler, RealSecure

1.3 OpenSource ips/ids systémy

1.3.1 Snort

Je v dnešnej dobe jeden z najrozšírenejších detekčných/prevenčných systémov. Jeho rýchle a široké rozšírenie je zapríčinené hlavne tým, že sa jedná o open-source nástroj, teda voľne šíriteľný software. Najväčšia výhoda je pri ňom implementácia naprieč operačnými systémami (Windows, Unix, Mac OS). Snort taktiež nezaostáva v pravidelnom vývoji, aktualizáciách a vylepšovaní. Taktiež výhodou je možnosť vlastnej konfigurácie systému, ako napríklad písanie vlastných pravidiel, vďaka čomu má užívateľ možnosť zredukovať falošné poplachy, alebo sa zamerať na určitý druh napadnutia. Obzvlášť falošné útoky sú veľmi nežiadúce a preto sa im vlastnou vhodnou konfiguráciou dá predchádzať. treba sa však zamyslieť aj nad tým, že nie je možné odbúranie všetkých falošných útokov pri variante, že bude sieť naplno chránená. Vždy je nutné si zvoliť vhodnú variantu konfigurácie. [19] [20]

Režimy systému Snort: V starších verziách systému snort pracoval len v režime "packet sniffer". Neskôr sa pri vývoji a vylepšovaní systém rozšíril o ďalšie dva režimy, packet logger a network intrusion detection system (NIDS).

1.3.2 Packet sniffer

Označovaný aj ako ploštica, sliedič či číhač. Pracuje na princípe, že zachytáva prechádzajúce pakety a vypisuje ich priamo na obrazovku. Podrobnejší popis je uvedený nižšie v texe.

1.3.3 Packet logger

Označovaný aj ako záznamník. Pracuje na podobnom princípe ako packet sniffer s rozdielom, že sa dáta zaznamenávajú na disk.

Princíp:

Packet sniffer a Packet logger vykonávajú v princípe rovnaké operácie. Zachytenie sieťovej premávky a následné spracovávanie paketov, či už sa jedná o zapisovanie na disk do súboru, alebo do obrazovej podoby na monitore. Obdobou je aj program wireshark alebo tcpdump (v základnej linuxovej výbave). Kvôli jednoduchosti sa väčšinou tieto režimy cez Snort nepoužívajú. Praktickejší a najviac využívaný režim Snortu je NIDS. [19] [20]

1.3.4 NIDS

Pri tomto režime Snortu sa vytvára triedenie všetkých paketov, ktoré sú následne analyzované a vyhodnocované podľa preddefinovaných pravidiel na rozradenie, ktoré pakety sú prvotne vnímané ako potencionálne útoky. S týmito následne pracuje. Týmto postupom sa docieli výrazného zníženia potrebného výkonu Snortu a tak isto pravdepodobnosť falošných poplachov. Snort obsahuje niekoľko zásuvných modulov (komponentov), ktorými pakety prechádzajú.

Celá sieťová prevádzka prechádza skrz zásuvné moduly procesora, ktoré rozhodujú o tom, ktoré pakety budú následne analyzované. Takéto rozhodovanie zabezpečuje tzv. packet classifier, ktorý pakety skúma a rozhoduje, či je paket typu napríklad Token Ring, či o ethernetový rámec, o aký typ protokolu sa jedná (UDP, TCP a pod.), či ide o IP alebo iný rámec, či je rámec súčasťou VLAN atď. Pri vyhodnotení paketu ako potenciálne nebezpečný, je ďalej skenovaný detekčnou jednotkou.[19] [20]

1.4 Surikata

Surikata je bezplatný open-source vyspelý, rýchly a robustný sieťový prostriedok na detekciu hrozieb. Suricata je schopná detekovať vniknutie v reálnom čase (IDS), zabezpečovať prevenciu narušenia (IPS), monitorovať bezpečnosť siete (NSM) a spracovávať offline pcap komunikáciu (dokáže teda zachytávať a ukladať sledovanú sieťovú komunikáciu a neskôr se k nej vráti za účelom jej ďalšej analýzy offline).

Suricata kontroluje sieťovú prevádzku pomocou silných a rozsiahlych pravidiel, sigantúr a má silnú podporu Lua skriptovania na detekciu komplexných hrozieb. Suricata je rýchlo sa rozvíjajúca komunita zameraná na vývoj. Zameriava sa na bezpečnosť, použiteľnosť a efektívnosť.

Projekt a kód Suricata vlastnia a podporujú Nadácia otvorenej informačnej bezpečnosti(Open Information Security Foundation [OISF]), nezisková organizácia, ktorá sa zaviazala zabezpečiť vývoj a trvalý úspech spoločnosti Suricata ako projektu open-source.[14] [15]

1.5 Sieťové nebezpečné aktivity

V sieťových technológiách sa stretávame s rôznymi nebezpečnými aktivitami. Kvôli lepšiemu prehľadu definujeme tieto aktivity následovne.

- Útok : pokus vniknutia do systému.
- Prienik : aktivita/ séria aktivít, ktoré predstavujú hrozbu pre bezpečnosť siete, zariadení a dát.
- Incident : prienik do systému predstavujúci napadnutie bezpečnosti systému, ktoré môže byť nejakým spôsobom charakterizované. [16]

1.5.1 Delenie útokov

Útoky vieme v základe rozdeliť na dve skupiny:

1. Pasívne- sústreďujú sa na získanie prístupu k sieti alebo systému bez toho, že by ohrozili dáta alebo sieťové zariadenia
2. Aktívne- sa tiež snažia získať prístup do siete, ale ohrozujú používateľove dáta, sieťové zariadenia a systém ako taký.

Ďalšia metóda rozdelenia internetových útokov je zameraná na vzťah medzu útočníkom a obeťami.

1. Interný útok- pochádza priamo z napadnutej siete. Väčšinou z lokálnej siete firmy alebo organizácie.
2. Externý útok- sú všetky ostatné, ktoré pochádzajú z vonku. Najčastejšie z internetu.[5]

1.6 Definovanie útokov za konkrétnym účelom

Každý z útokov je uskutočňovaný s nejakým konkrétnym zámerom. Pri IDS systéme sa definujú tri druhy týchto útokov.

1.6.1 Útoky s cieľom dosiahnuť prístup do systému

Tieto útoky majú primárnu úlohu sa nabúrať do systému a získať dôležité informácie. Následne má útočník k dispozícii ďalšie možnosti a akcie. Medzi známe patria: trójske kone, skenovanie komunikácie,portov, IP adries či služieb, útoky na prelomenie hesiel, krádež dokumentov a citlivých dát, vytváranie internetových spojení a pod. Dôležité sú však hlavne „Identity spoofing“, „Man in the Middle“a „Social engineering“. [16]

- **Identity spoofing(IP spoofing)** typ útoku, pri ktorom osoba alebo program maskuje svoju totožnosť a tvári sa ako druhá osoba. T.j. útočník sfalšuje známu,platnú IP adresu, na získanie prístupu do inej privátnej siete. [16]
- **Man in the middle** útok, ktorého podstatou je snaha útočníka odpočúvať komunikáciu medzi účastníkmi tak, že se stane aktívnym prostredníkom. [16]
- **Social engineering** spôsob manipulácie ľudí za účelom vykonania určitej akcie alebo získania určitej informácie. Tento útok sa spolieha na interakciu s užívateľom a často klame užívateľa, aby boli porušené štandardné bezpečnostné procedúry. **Príklady:** vymyslenie scenára s cieľom presvedčiť obeť k činom, ktoré vedú k prezradení informácie. Ide väčšinou o skĺbenie nepravdivej s čiastočne pravdivou informáciou získanou skôr. Môže ísť o získavanie rodných čísel, dátumu narodenia, meno nadriadeného, číslo bankovej karty a podobne. Praktiky sa hýbu od klasického súkromného telefonátu do sídla firmy, až po maily, ktoré následne presmerujú na útočníkovú stránku a vyzývajú k zadaní chúlostivých informácií. [16] [17]

1.6.2 Útoky s cieľom zmeny v systéme

Tieto útoky slúžia na získavanie, zmenu alebo odstraňovanie dát. Útočníci mnohokrát získavajú prístup do systému. Útoky teda môžu byť prostriedkom k získaniu autorských práv v systémoch, alebo falšovania identity. Najčastejšie sa týmto druhom útokov upravujú firemné databáze.

1.6.3 Útoky s cieľom odoprenia služieb (Denial of Service)

Tento typ útoku používa veľké množstvo zbytočných informácií, ktoré zahlcujú systémy a tým blokujú komunikáciu a služby. Následkom toho je buď vynútenie viacnásobného reštartu systému, alebo čiastočné či úplne zahltenie komunikácie medzi účastníkmi komunikácie. Najčastejšie sa jedná o **zahltenie žiadosťami o odozvu** tzv. **ICMP floods**, kde sa prijímajú všetky žiadosti typu ping a systém je nútený na každú odpovedať formou ICMP paketu. **Peer-to-peer** útok, kde ide o žiadosti o prijatie do P2P siete. **Teardrop** zahŕňa posielanie IP fragmentov s prekrývajúcim sa veľkým objemom dát na cieľový počítač. [5]
Distributed Denial of Service - DDoS Tento útok je charakterizovaný väčším počtom počítačov, snažiacich sa naraz zahltiť cieľ útoku. Veľakrát je útok uskutočňovaný bez vedomia majiteľov počítačov. Jeden zo spôsobov je malware, ktorý má v sebe pevnú IP adresu obete a dátum, kedy sa program pokúsi na cieľ zaútočiť. Po infikovaní počítača nie je nutné aby útočník musel komunikovať s daným systémom a útok prebehne automaticky. Taktiež môže byť systém napadnutý programom

(botom), ktorý potom beží ako proces, ktorý čaká na príkazy od útočníka. Takto napadnutý systém potom označujeme ako zombie. Zoskupenie zombie napadnutých tým istým programom potom nazývame botnet. [**?**]

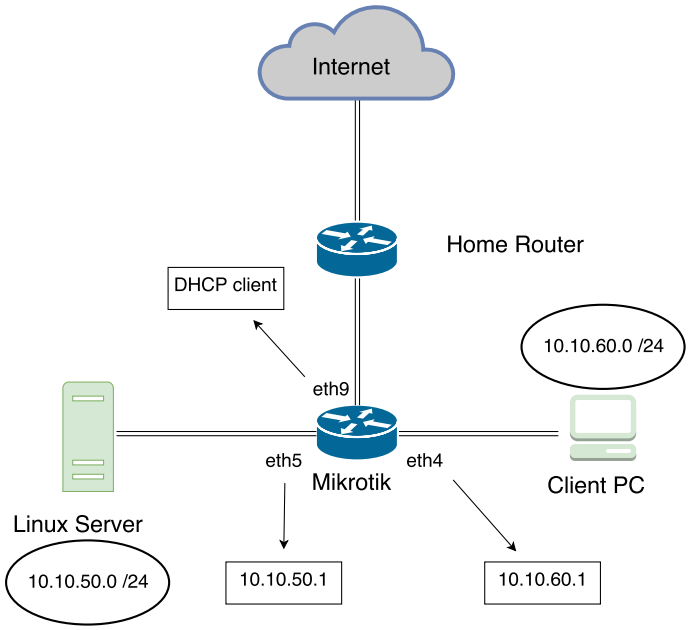
1.7 Detekcia útokov

V prvom rade je nutné si uvedomiť, že sa dnes stretávame s veľmi dobre sa maskujúcimi druhmi internetových útokov. Dôkladnejšie skúmanie jednotlivých útokov, ako aj ich prejavy je nutné nezanedbať. Medzi hlavne ukazovatele napadnutia môžeme zaradiť zníženie rýchlosti pripojenia k internetu, zmeny v sieťovej komunikácii, nesprávne fungovanie programov a iné. Medzi hlavné pozorované atribúty patrí kontrola IP adresy paketov prichádzajúcich z internetu do lokálnej siete tváriacich sa ako by prichádzali z lokálnej siete(IP spoofing). Ďalším atribútom je abnormálna aktivity. Ide o útočenie na prienik do systému, pričom vykonávanie útoku sa väčšinou opakuje. Zabránenie je možné stanovením hranice počtu pokusov o prihlásenie za určité časové obdobie a podobne. Obchádzanie detekcie býva taktiež možné, pomocou úpravy sieťových nastavení. Fragemntácia paketov tiež býva problematická, pretože útočníci pomocou tejto metódy maskujú svoje útoky. Detekcia upraveného paketu je zložitejšia, pretože pakety sa na prvý pohľad neškodné. [5]

2 PRAKTICKÁ ČASŤ

2.1 Príprava

Na prvom mieste je nutné si spraviť rozpis a prípravu, ako by mal systém vyzerať a ako sa správať. Na následujúcom obrázku je vidieť celkové prepojenie systému.



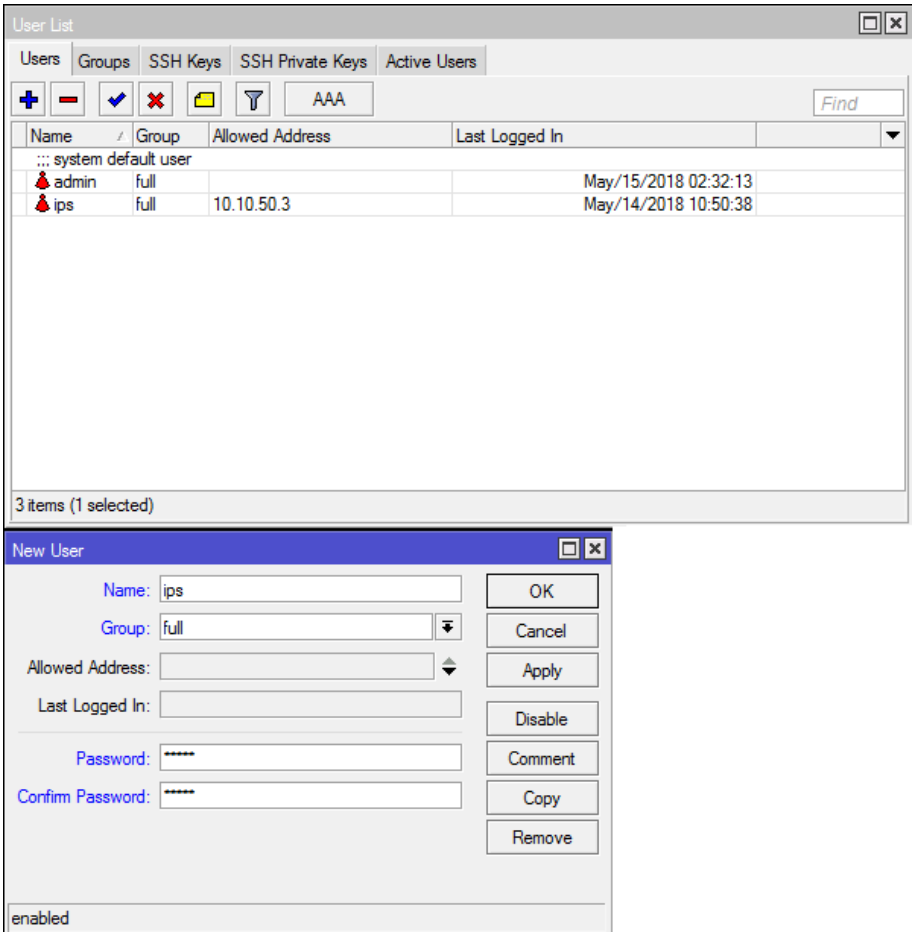
Obr. 2.1: Topológia

2.2 Inštalácia Mikrotiku

Na stránkach www.mikrotik.com pod záložkou Downloads je možné vyhľadať najnovšiu verziu RouterOS podľa verzie RouterBoardu. V tomto prípade je to verzia 6.40.5 a stiahnutý súbor má názov: „routeros-mipsbe-6.40.5“. Stiahnutý RouterOS následne jednoduchým pretiahnutím nakopírujeme za pomoci WinBoxu do **Files**. Po ďalšom reboote je potom možné vidieť novú verziu routerOS. Následne som spravil prvotné sieťové nastavenie. Priradenie portov, DHCP server na LAN, DHCP clienta na WAN (pretože testované prostredie je v za domácim routerom a tak sa Mikrotik tvári ako klientská stanica, ktorá obdrží IP adresu), Linux server dostal pevne stanovený rozsah(na testovacie účely) a pripája sa statickou IP adresou. Pre WAN port je potrebné nastavenie IP-Firewall-NAT -> Chain: srcnat, Out.Interface: WAN a Action: masquerade.

SSH pripojenie na mikrotik

Aby som sa mohol pripojiť na mikrotik cez SSH službu, musel som vytvoriť užívateľa s heslom, pričom pre zabezpečenie som mu priradil konkrétnu IP adresu z ktorej sa môže pripájať. Toto je potrebné pre skript, v ktorom definujem meno a heslo pre pripojenie a následné zaslanie IP adresy útočníka na zariadenie mikrotik. Do zaria-



Obr. 2.2: Pridanie užívateľa do mikrotiku

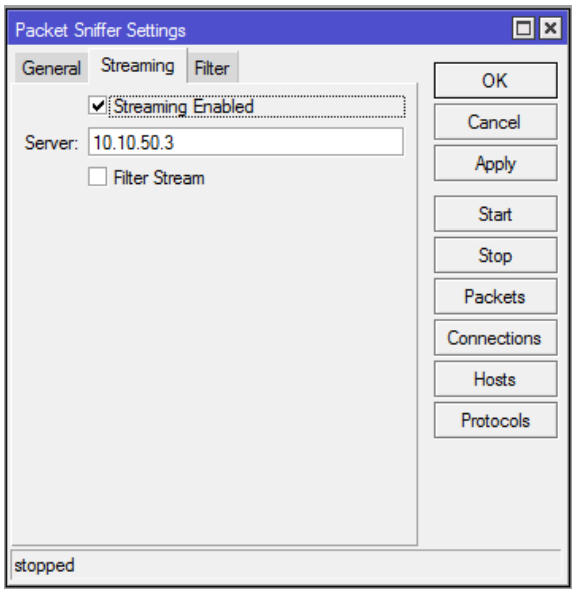
denia mikrotik bolo taktiež nutné pridať jedno pravidlo, na ktoré sa bude odkazovať skript a ktorým sa budú zapisovať IP adresy útočníkov do „address listu“, konkrétne do blacklistu.

```
/ip firewall filter add src-address-list = blacklist action=drop
```

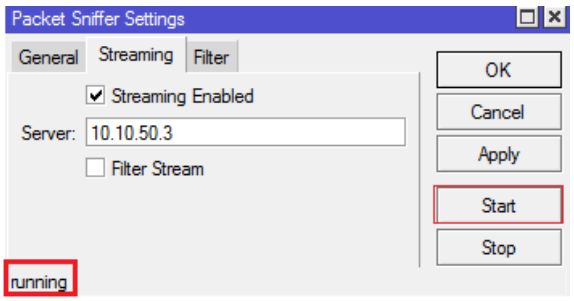
2.3 Packet sniffer

Je nástroj, ktorý dokáže zachytávať a analyzovať pakety, ktoré sa chystajú, opúšťajú alebo prechádzajú cez mikrotik(s výmnimkou prevádzky, ktorá prechádza len

cez switchovací čip). Pre správne fungovanie bol najskôr nainštalovaný „package Calea“a následne otvorený nástroj **Packet Sniffer**. V nastaveniach tohoto nástroja pod záložkou **Streaming** je treba zadať IP adresu linuxového servera, na ktorú sa všetky požadované pakety preposielajú. V záložke **Filter** sa udáva z rolovacieho okna port ktorý chceme odchytávať a preposielať. Nastavenie smeru sme vybrali RX, pretože chceme zachytávať prichádzajúcu komunikáciu z WAN portu.

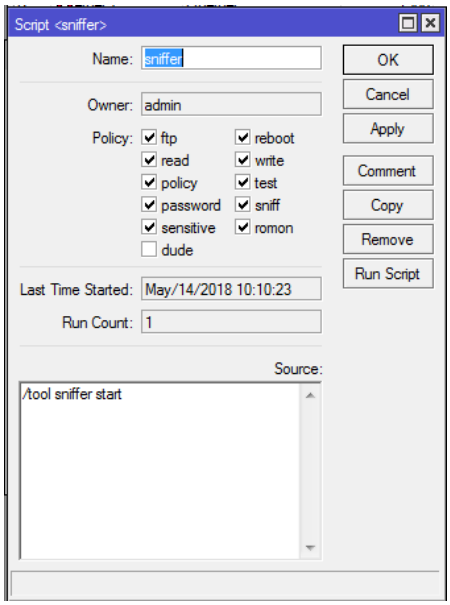


Obr. 2.3: Nastavenie Packet Sniffera

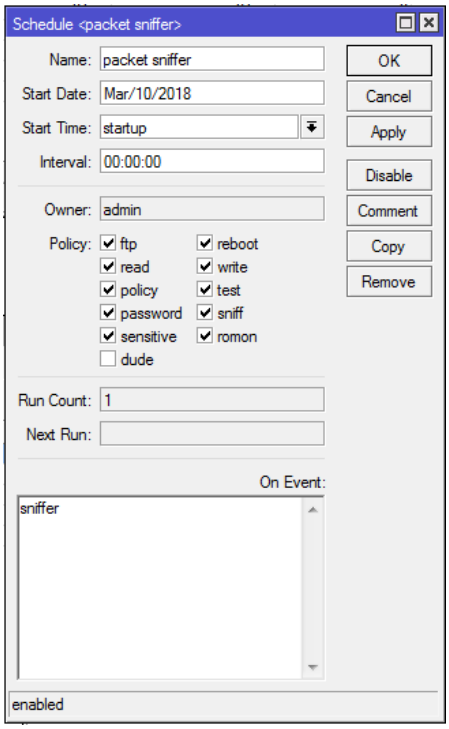


Obr. 2.4: Spustenie a indikácia

Po stalčení tlačidla štart sa zmení status zo Stopped na Running. V tomto momente sa na mikrotiku vytvára nová komunikácia, ktorá prichádza na zadefinovanú IP adresu: 10.10.50.3 v zakódovanom tvare. Pakety sú balené prostredníctvom TZSP(TaZmen Sniffer Protokol) a posielané cez UDP protokol. [22] Na mikrotiku som vytvoril skript(System->Scripts) pre spúšťanie Packet Sniffera, ktorý som v plánvači(System->Scheduler) nastavil na automatické spúšťanie po štarte systému RouterOS.



Obr. 2.5: prikaz na spustenie Packet Sneffera



Obr. 2.6: Planovač spustania Packet Sniffera

2.4 Konfigurácia Linux Servera

2.4.1 Prvotná konfigurácia

Vybral som 2 vyrianty dostupných, odladených verzií linuxovej distribúcie Ubuntu 16.04.3 LTS Xenial a Ubuntu 18.04. Pre správne fungovanie neskôr inštalovaného Snortu je najskôr potrebné nainštalovať balíčky:

- **build-essential** - poskytuje nástroje na zostavovanie nástrojov a kompiláciu softvéru.
- **bison, flex** - analyzátor vyžadovaný DAQ(kt. je inštalovaný neskôr).
- **libpcap-dev** - knižnica na zachytávanie sieťovej komunikácie.
- **libpcre3-dev** -knižnica na podporu regulárnych výrazov.
- **libdumbnet-dev** - statická knižnica pre dáta.
- **zlib1g-dev** - kompresná knižnica.
- **liblzma-dev** - poskytuje dekompresiu súborov typu swf(adobe flash)
- **openssl, libssl-dev** - poskytuje podpisy súborov MD5 a SHA
- **libcrypt-ssleay-perl** - podpora OpenSSL pre LWP
- **liblwp-useragent-determined-perl** - používateľský agent WWW knižnice jazyka Perl. Pridáva do LWP toleranciu voči chybám tým, že opakuje pokus pri výskyte chyby.
- **libnghttp2-dev** - poskytuje vývojovú knižnicu pre Nghttp2: knižnica HTTP/2C, ktorá implementuje algoritmus kompresie hlavičky HPAC. Pri nižšej verzii Ubuntu je proces inštalácie zložitejší.

Realizované príkazom:

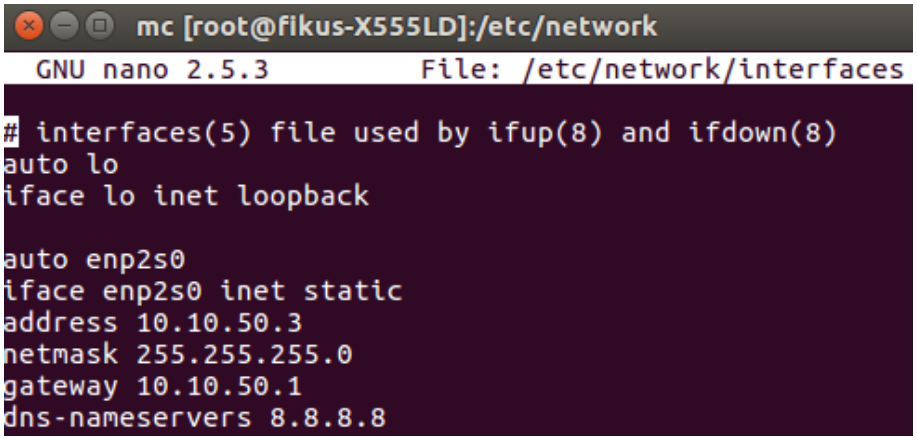
```
root@fikus:~# apt-get install -y build-essential bison flex libpcap-dev
 libpcre3-dev libdumbnet-dev zlib1g-dev liblzma-dev openssl libssl-dev
 libcrypt-ssleay-perl liblwp-useragent-determined-perl libnghttp2-dev
```

Následne bola stiahnutá posledná verzia DAQ(Data AcQuision library) zo stránky: „https://www.snort.org/downloads“konkrétne verzia „daq-2.0.6.tar.gz“. Rozbalenie a nainštalovanie prebehlo príkazmi:

```
root@fikus:~# snort_src/
root@fikus:~/snort_src# tar -xvzf daq-2.0.6.tar.gz
root@fikus:~/snort_src# cd daq-2.0.6
root@fikus:~/snort_src/daq-2.0.6# ./configure
root@fikus:~/snort_src/daq-2.0.6# make
root@fikus:~/snort_src/daq-2.0.6# make install
```

2.4.2 Nastavenie statickej IP adresy

Musel som nastaviť statickú IP adresu, aby bolo neskôr možné smerovanie komunikácie na konkrétnu adresu. Pri Ubuntu 16.04.03 som sa potýkal s nefunkčnosťou grafického rozhrania a teda som pristupoval k zmene IP adresy cez úpravu súboru „interfaces “. S cestou /etc/network/interfaces.



Obr. 2.7: Nastavenie IP adresy pre Linux server

Zmena sa uloží až po reštarte služby,a lebo reboote systému.
Pri Ubuntu 18.04 problém s grafickým prostredím nie je a tak je možné nastaviť statickú IP adresu cez grafické rozhranie v nastaveniach sieťovej karty. Po tomto kroku sme pripravený na samotnú inštaláciu Snortu.

2.4.3 Inštalácia Snortu

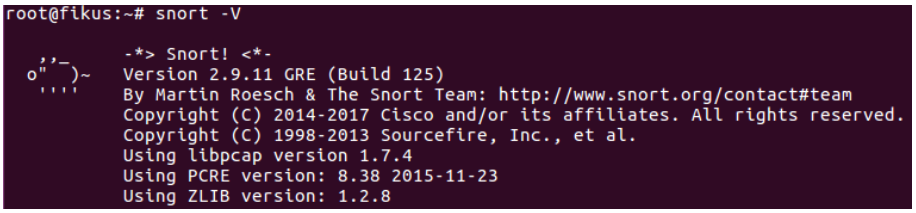
Rovnako ako DAQ som zo stránky „https://www.snort.org/downloads“stiahol potrebný súbor konkrétne „snort-2.9.11.tar.gz“. Súbor som rozbalil a nainštaloval rovnakými príkazmi ako DAQ v rovnakom priečinku (snort_src). Pri inštalácii som zadal HOME_NET IP adresu, keď som vyzvaný, 10.10.50.0 s prefixom 24.

Obr. 2.8: snort_src priečinok

Spustením príkazov nižšie sa updatujú zdielané knižnice a pridá sa väzba do binárnych súborov.

```
root@fikus:~/# ldconfig
root@fikus:~/# ln -s /usr/local/bin/snort /user/sbin/snort
```

Po predchádzajúcich krokoch(inštalácie) môžeme otestovať, či nám Snort pracuje. Použijeme na to príkaz snort s parametrom -V, ktorý nám zobrazí aj verziu.



Obr. 2.9: snort -V

2.4.4 Konfigurácia Snortu

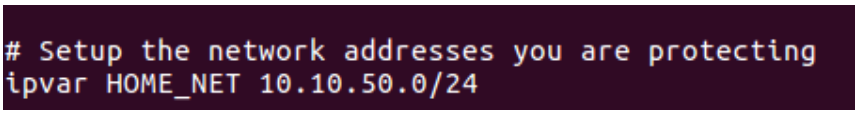
Keďže ide o bezpečnosť, tak chceme aby as Snort spúšťal v neprivilegovanom režime. To zaistíme vytvorením užívateľa a aj skupinu s názvom **Snort**.

```
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT\_IDS -g snort
```

Pri vytváraní Snoru ako NIDS je potrebné pridať niekoľko súborov a následne zmeniť vlastníctvo na novo vytvoreného užívateľa. Podľa tabuľky v prílohách.A.1

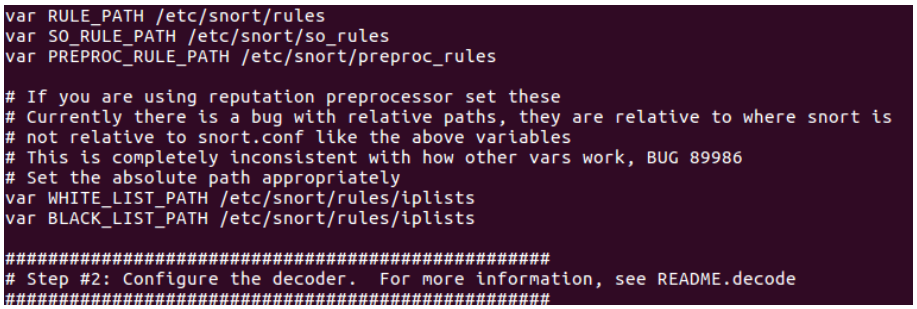
Nutné je vytvoriť priečinky a súbory pre umiestnenie a zadávanie konkrétnych pravidiel a toku dát. Pri inštalácii sa všetky potrebné priečinky a súbory neaplikovali a nenahrali kam mali, musel som ich preto nakopírovať ručne z rozbaleného stiahnutého súboru „snort-2.9.11.tar.gz“A.2

Priama konfigurácia sa deje v konfiguračnom súbore s označením snort.conf, do ktorého sa dostaneme cestou /etc/snort/. Pri otvorení tohto súboru niektorým z textových editorov(vi, nano ...) meníme konfiguráciu prispôsobením na naše prostredie. Na 45. riadku nastavíme za „ipvad HOME_NET“nastavíme IP adresu na ktorej budeme detekovať hrozby. Z vytvorenej topológie, zaznamenanej v Obr.2.1 je zrejmé že ide o sieť s IP adresou a prefixom: „10.10.50.0/24“. Na 104. riadku



Obr. 2.10: 45.riadok

a následujúcich sú vypísané cesty k pravidlám a iplistom. Potrebné je prispôsobiť tieto cesty aby sedeli s aktuálnou štruktúrou. Na 545.riadku som odkomentoval zahrnutie lokálnych pravidiel jednoduchým zmazaním znaku „(#)“. Potom som pred vytváraním pravidiel otestoval, či sú dáta upravené správne a či sa Snort spustí bez

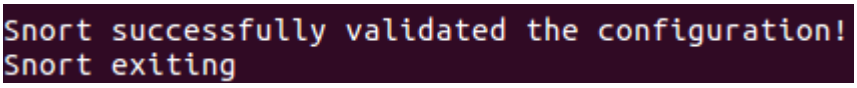


Obr. 2.11: 104.riadok až 114.riadok po úprave

chýb. Príkaz zahŕňa testovanie, pričom na konci príkazu je potrebné použiť názov rozhrania sieťovej karty v mojom prípade „enp2s0“.

```
root@fikus:~# snort -T -i enp2s0 -c /etc/snort/snort.conf
```

Úspešná validácia konfigurácie vypíše do konzoly správu spolu s výpisom konfigurácie. Zobrazí sa veľa parametrov, ktoré je dobre vidieť, no najhlavnejším je tabuľka s výpisom počtov pravidiel použitých pre konkrétny protokol: ICMP, UDP, TCP, IP.



Obr. 2.12: Úspešná validácia

2.4.5 Prvé pravidlo pre test Snortu

V predchádzajúcej pod-kapitole sme si odkomentovali zahrnutie súboru lokálnych pravidiel **local.rules** v konfiguračnom súbore **snort.conf**. Súbor lokálnych pravidiel si v tejto kapitole nastavíme tak, že bude obsahovať jedno pravidlo ktoré bude zahŕňať ICMP pakety. Súbor s cestou /etc/snort/rules/local.rules som otvoril textovým editorom a napísal nasledujúce pravidlo:

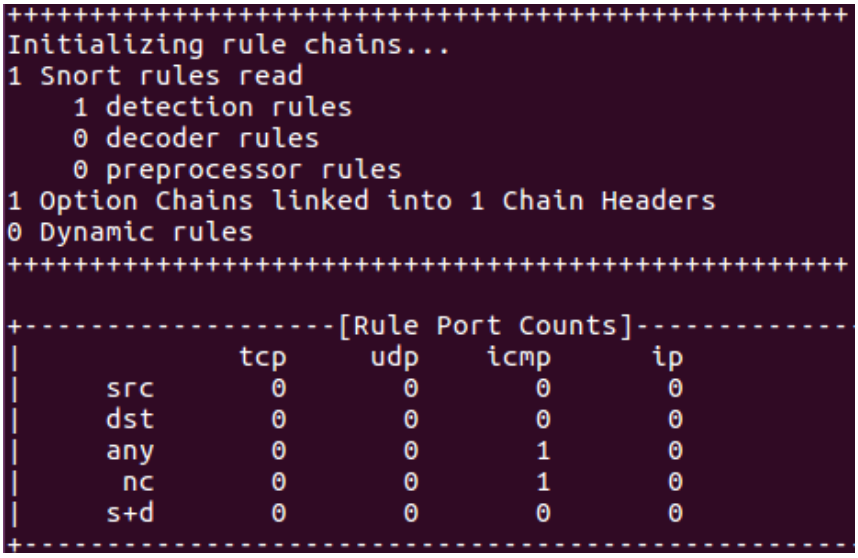
```
alert icmp any any -> \$HOME_NET any (msg:"ICMP test detected";
GID:1; sid:10000001; rev:001; classtype:icmp-event;)
```

Toto pravidlo hovorí, že ak príde ICMP paket z akejkoľvek IP adresy na akúkoľvek adresu z našej HOME_NET adresy(10.10.50.0/24), generuje správu alert so správou „ICMP test detected“pričom ďalšie informácie, ktoré sa vypisujú, sú dôležité pri skupinách pravidiel, na rozlíšenie a manažovateľnosť.

Po pridaní a vložení nového pravidla je potrebné si znovu zvalidovať konfiguráciu snortu. Po zadaní príkazu pre validáciu:

```
snort -T -i enp2s0 -c /etc/snort/snort.conf
```

sa nám zobrazila rovnaká hláška, s tým rozdielom, že v tabuľke pravidiel pribudlo pravidlo ICMP.



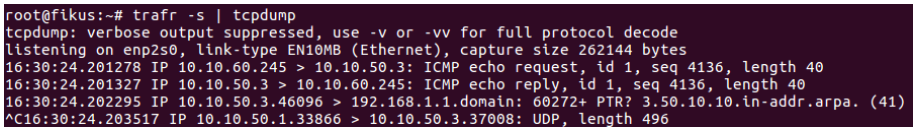
Obr. 2.13: Výpis pravidiel

2.4.6 Test pravidiel

Testovať, či prebehlo všetko správne môžeme až po tom, čo dekódujeme dáta. Keďže sú pakety prechádzajúce cez Mikrotik zakódované protokolom TZSP, preto je nutné použiť **nástroj Taft**, aby sa pakety dekódovali na úrovni Linuxového servera. Tento nástroj je voľne dostupný na stránkach Mikrotiku, odkiaľ som ho stiahol, rozbalil a súbor s názvom „trafr“som presunul do „/usr/local/bin“medzi spustitelné programy. Pri 32 bitových verziách je treba doplniť balíček „libc6:i386 “príkazom [25]:

```
apt-get install libc6:i386
```

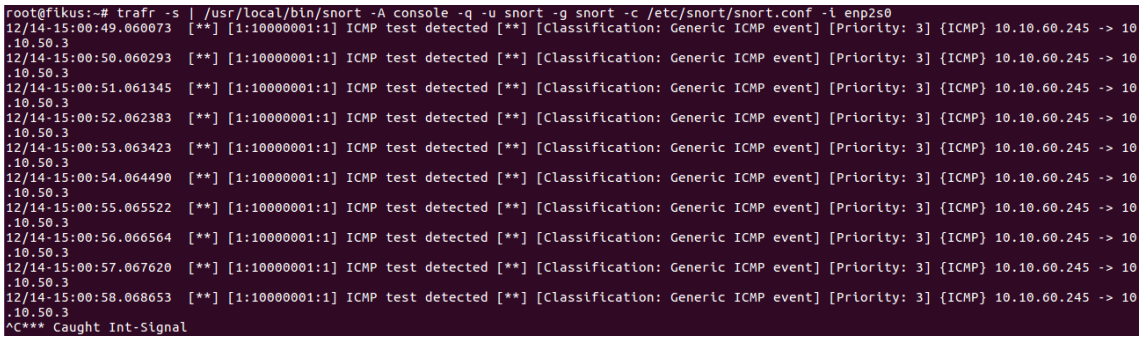
Prvý test som vykonal aby som zistil, či Traft prijíma pakety a či sú v poriadku.



Obr. 2.14: Test Traftu

Týmto som sa uistil, že prijímanie a dekódovanie paketov prebehlo v poriadku. Akonáhle som zistil že Traft funguje a vykonáva svoju činnosť, mohol som otestovať, či bude fungovať aj v spolupráci so snortom. test prebehol zadaním príkazu:

```
trafr -s | /usr/local/bin/snort -A console -q -u snort -g snort -c
/etc/snort/snort.conf -i enp2s0
```



Obr. 2.15: Traft so Snortom

Dekódované(Traftom) a detekované(Snortom) pakety sa nám týmto pádom zapisujú do konzoly. Ďalším krokom je vytvoriť spoluprácu medzi Traftom a Snortom, pričom je nutné, aby sa log zaznamenával. Tento krok vieme vyriešiť jediným príkazom, kde použijeme následovné parametre:

- **-s** program zapisuje do syslogu
- **-c** konfiguračný súbor aj s cestou
- **-l** cesta, kam má Snort logovať

```
trafr -s | snort -c /etc/snort/snort.conf -l /var/log/snort
```

Tento príkaz teda slúži na dekódovanie a spracovávanie prijatých paketov a následné zapisovanie(vyhodnocovanie) do logu v zložke „/var/log/snort/“konkrétne do zložky „**alert**“.

2.4.7 Zmena smerovania hlásení o útokoch

Pre fungovanie systému podľa predstáv je potrebné smerovať hlásenia o útokoch do súboru **syslog** s cestou „/var/log/syslog“. V konfiguračnom súbore snortu je preddefinovaná funkcia pre priame zapisovanie[24]. V konfiguračnom súbore je treba teda odkomentovať jednoduchým zmazaním mriežky(#) následujúci riadok:

```
output alert_syslog LOG_AUTH LOG_ALERT
```

Toto však pre Ubuntu 16.04 a ani 18.04 na ktorých pracujeme nestačí. Ďalej sa treba zamerať na zložku „/etc/rsyslog.d“, kde je potrebné vytvoriť konfiguračný súbor s ľubovolným názvom. Tento súbor sa bude potom spúšťať automaticky, pretože v **rsyslog.conf** je zapísané, že má všetky konfiguračné súbory zo súboru

„/etc/rsyslog.d“zahrnúť (funkcia $IncludeConfig). V mojom prípade som vytvoril súbor **auth.conf** a do neho som dopísal kde sa nachádza syslog následovne:

```
auth /var/log/syslog
```

Testovanie zapisovania som urobil podobne ako pri testovaní zapisovania do zložky **alerts**. Príkazom pre zapisovaní snrotu do zložky **alerts** s pridaním jedného parametra (**-s**) som spustil spoluprácu Traftu so snortom:

```
trafr -s | snort -s -c /etc/snort/snort.conf -l /var/log/snort
```

Následne som spustil ping z iného PC na adresu 10.10.50.3 pre simulovanie útoku. Do syslogu sa mi zapísala následujúcu sprava:

```
May 13 15:30:30 fikus-X555LD snort: [1:1000002:1] ICMP connection attempt portscan {ICMP}
10.10.60.254 -> 10.10.50.3
```

Obr. 2.16: Syslog alert

2.5 Skripty

Na vytvorenie IPS systému z aktuálneho NIDS je potrebné vytvoriť PHP skript, ktorý som vytváral s pridržaním sa stránok Mikrotiku[18]. Skript obsahuje prehľadávanie syslogu, kde kontroluje, či za posledných XX sekúnd/minút Snort nezaznamenal útok. Ak v syslogu útok nájde podľa parametrov zadaných pre vyhľadávanie v PHP skripte, otvorí cez SSH službu spojenie a spustí skript na strane mikrotiku. PHP skript na strane Linuxu: A.4
Na strane mikrotiku je potrebné vytvoriť skripty pre zaznamenanie času a ukladanie IP adresy do blacklistu. Na základe pravidla pre blacklist sa potom táto IP adresa blokuje. Skripty na strane mikrotiku: A.5 A.6

2.5.1 Skript na Linuxe

Vytvoril som PHP skript (php.php a presunul do zložky /root) v ktorom sa na začiatku definuje odkiaľ chceme vyberať jednotlivé záznamy o útokoch. Záznamy sú vyberané na základe času, ktorý dokážeme meniť (sekundy či minúty). V mojom prípade som nastavil čas na 30 sekúnd. Vďaka tomuto času môžeme teda sledovať syslog záznamy za posledných 30 sekúnd. Následne sa za pomoci ďalších príkazov prehľadáva syslog riadok po riadku a hľadá záznamy obsahujúce výrazy: „Priority:

“a „portscan“. PHP skript ďalej pokračuje regulárnym výrazom, ktorý v konkrétnom riadku hľadá IP adresu a následne ju prehodí do premennej „filter “. Následne sa priradí nájdená IP adresa do poľa „blocked “, ak sa v poli ešte nenachádza. Pokračuje sa kontrolou IP adries, ktoré sú tzv. bezpečné (IP adresy z LAN siete kde sa nachádza linux server s programom snort). V ďalšom kroku sa vykoná funkcia sendMikrotik v ktorej je udaná adresa mikrotiku, kde naväzujeme komunikáciu cez ssh s používateľským menom a heslom pre naviazanie tejto komunikácie. A.4

2.5.2 Skripty na Mikrotiku

Vytvoril som následne skripty na mikrotiku (cez gui v záložke System-> Scripts, cez terminál -> system script add name=XXXXX a následne editovať -> system script edit XXXXX source).

- Prvý skript (time) pri jeho spustení vyťahuje aktuálny čas na mikrotiku a pričíta k nemu 5 minút(takto dlho zostane zapísaný záznam v blackliste) a následne vykoná zápis IP adresy do blacklistu s komentárom, kde je udaný čas s pridanými piatimi minútami.A.5
- Druhý skript (filter)na základe aktuálneho času prezerá a porovnáva, či v tabuľke blacklist nie je zápis, ktorého comment hodnota nie je menšia(prešlo viac času ako päť minút) ako aktuálny čas. Ak v komentári nájde čas ktorý je menší ako aktuálny, záznam vymaže a daná IP adresa, ktorá tam bola zapísaná a odmietaná je opäť dostupná.A.6

2.5.3 Crontab a automatické spúšťanie

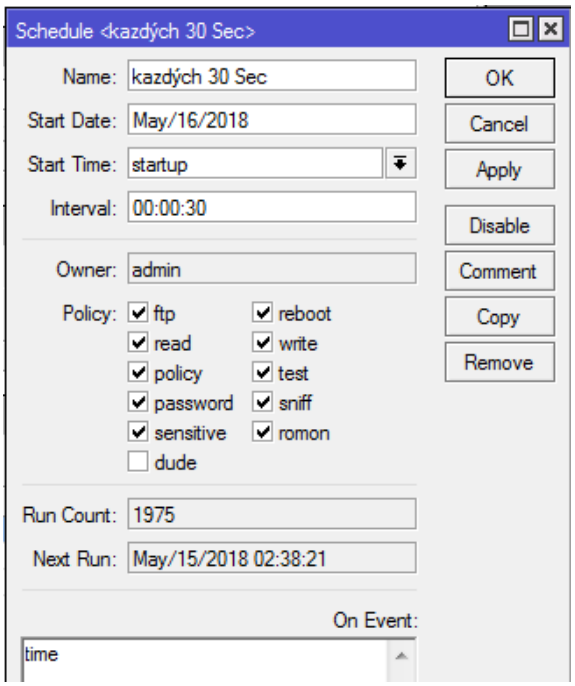
Pre automatické spúšťanie skriptu na linuxe som použil crontab, čo je vlastne nástroj, ktorý obsahuje plán vecí ktoré sa majú spustiť v určený čas. Tento nástroj je možné nastaviť, aby spúšťal príkazy v určitom časovom intervale (minútovo, hodinovo...), pričom najmenší časový údaj opakovaného spúšťania je jedna minúta.[23] Pre potreby rýchleho zapisovania IP adries a odbúrania útoku som teda musel zabezpečiť spúšťanie skriptu viac ako jeden krát za minútu. Zabezpečil som to dvakrát spustením toho istého skriptu s pridaním oneskorenia o 30 sekúnd. Do nastavenia úloh pre crontab som sa dostal jednoduchým príkazom:

```
crontab -e
```

```
*/1 * * * * php /root/php.php
*/1 * * * * (sleep 30 ; php /root/php.php)
```

Obr. 2.17: Crontab zápis pre spúšťanie každých 30s

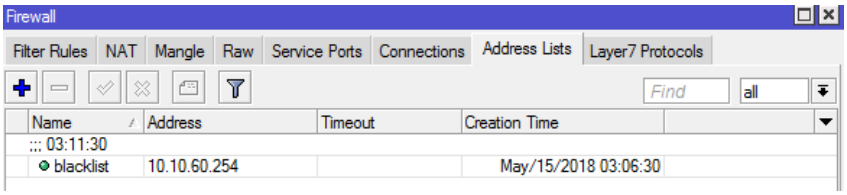
Pre automatické spúšťanie skriptu na zariadení Mikrotik som vytvoril v plánovači (System-> Scheduler) novú úlohu, ktorá bude spúšťať skript „time“každých 30 sekúnd.



Obr. 2.18: Mikrotik plánovač pre spúšťanie skriptu každých 30s

2.6 Test odstraňovania ICMP paketov a blokovanie IP adresy

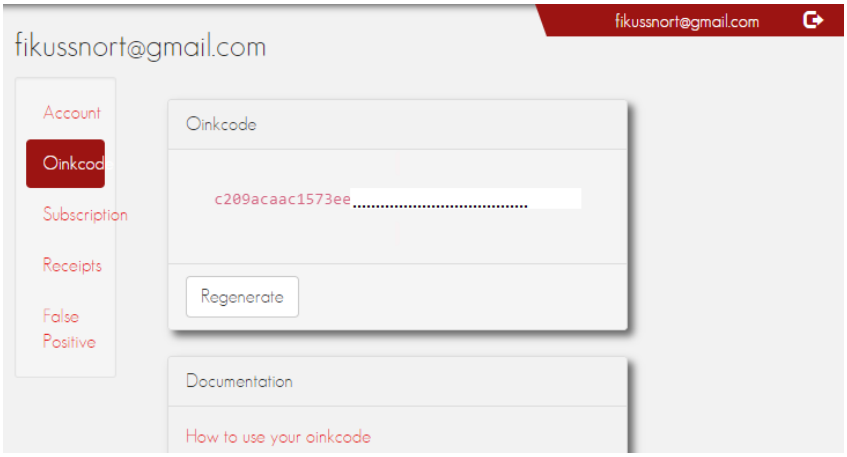
Pri tomto teste ide hlavne o fungovanie systému, ktorý vyhodnotí ping ako útok a následne zapíše záznam do syslogu, odkiaľ sa vďaka PHP skriptu vytiahne IP adresa počítača z ktorého vykonávame ping a následne ju pošle cez SSH spojenie do mikrotiku. Mikrotik túto IP adresu následne pridá do blacklistu a tá sa zablokuje na päť minúť. Po prejdení piatich minút sa IP adresa na základe skriptu vymaže z blacklistu a ping znova dostane odpoveď.



Obr. 2.19: Výsledok merania zavŕšené blokovaním IP adresy

2.7 Pulledpork

Pulledpork, je Cisco projekt pre snort, ktorý vytvára perl skripty na automatické sťahovanie a obnovovanie pravidiel pre Snort. Pre funkčnosť a načítanie pravidiel je potrebné mať tzv. „Onikcode“. Tento kód je možné nájsť po registrácií na stránkach „snort.org“. V hornej časti okna sa zobrazí emailová adresa a po kliknutí na ňu sa otvorý panel nastavení kde jedna z možností je Oinkcode:



Obr. 2.20: Výsledok merania zavŕšené blokovaním IP adresy

2.7.1 Inštalácia

Pre stiahnutie a konfiguráciu som použil následovné príkazy, pričom bolo treba doplniť aj niektoré ďalšie balíčky pre funkčnosť:

```
apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl
```

Stiahnutie a nastavenie umiestnenia:

```
cd ~/snort_src

wget https://github.com/shirkdog/pulledpork/archive/master.tar.gz -O
pulledpork-master.tar.gz

tar xzvf pulledpork-master.tar.gz
cd pulledpork-master/
cp pulledpork.pl /usr/local/bin
chmod +x /usr/local/bin/pulledpork.pl
cp etc/*.conf /etc/snort
```

Po tomto som otestoval, či je program pulledpork nainštalovaný a s tým zistil aj jeho verziu príkazom:

```
/usr/local/bin/pulledpork.pl -V
```

pričom výpis pulledporku bol: PulledPork v0.7.4 .

2.7.2 Úprava konfiguračného súboru

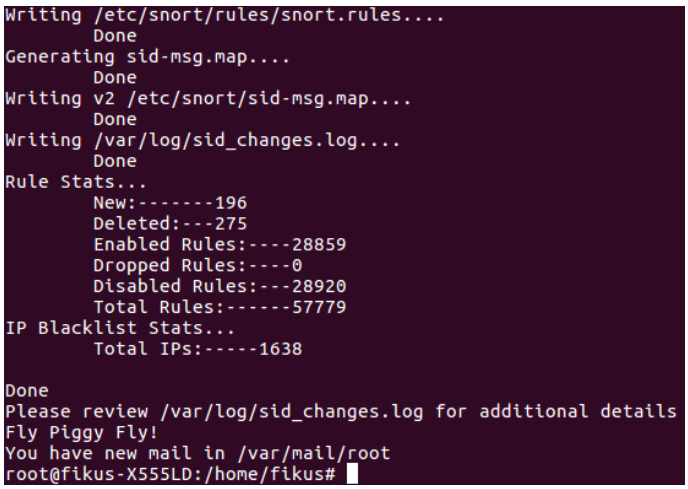
Pre konfiguráciu súboru som prešiel do konfiguračného súboru a zmenil som následujúce veci na konkrétnych riadkoch: A.7 Takto som zabezpečil, aby sa aj môj Oinkcode dostal pri sťahovaní pravidiel na stránky snortu a mohli sa bez problému stiahnuť.

Stiahnutie pravidiel

Pre stiahnutie pravidiel po úprave konfiguračného súboru bolo treba trab zapísať príkaz:

```
/usr/local/bin/pulledpork.pl -c /etc/snort/pulledpork.conf -l
```

Týmto sa PulledPork spustí a prebehne bez chyby, pričom posťahuje viac ako 57000 pravidiel z toho povolených(odkomentovaných) ich je necelých 29000. Po stiahnutí



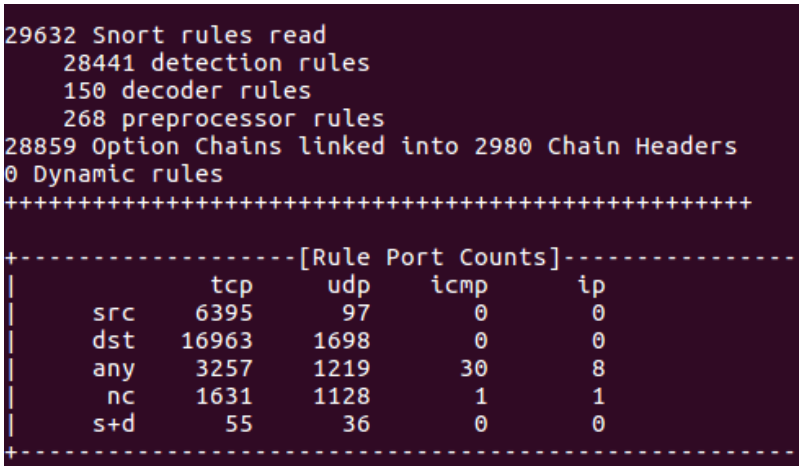
Obr. 2.21: Pridané pravidlá od pulledporku

pravidiel je teraz možné vidieť súbor s názvom snort.rule v adresári /etc/snort/rules. Pre úplne obsiahnutie pravidiel v programe snort je nutné povoliť túto sadu pravidiel. To nastavíme v konfiguračnom súbore snortu v adresáry /etc/snort/ a zložke snort.conf. Riadok číslo 547, hneď za riadkom, ktorý je treba zakomentovať(zadaním

znaku: “#“na začiatku riadku) aby sa nám pravidla nemiešali, je nutné odkomentovať(zmazaním znaku“#“). Po úprave konfiguračného súboru som sa presvedčil, že sa tieto pravidlá zaznamenali aj pri spúšťaní programu snort. Snort som znova spustil príkazom:

```
trafr -s | snort -s -c /etc/snort/snort.conf -l /var/log/snort
```

Počas spúšťania snortu sa mi zobrazila v príkazovom riadku tabuľka , kde je možné vidieť, koľko pravidiel teraz snort zahŕňa.



Obr. 2.22: Pravidlá v snorte

Automatické sťahovanie pravidiel

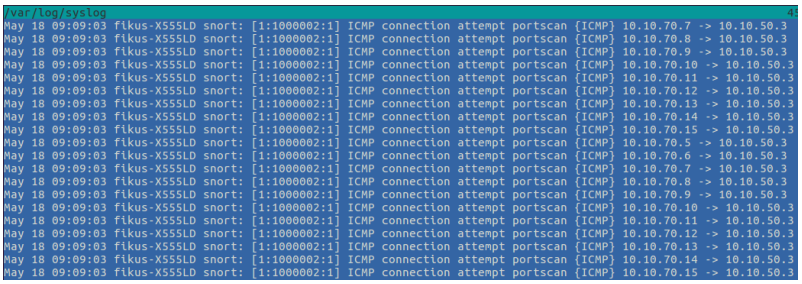
Pre automatické sťahovanie pravidiel je potrebné mať upravený konfiguračný súbor PulledPorku podľa predlohy vyššie v texte. Čiže je nutné definovanie minimálne zložky s pravidlami(rule_file), oinkcode, cesta odkiaľ sa ťahajú pravidlá(temp_path) a cestu k pravidlám pre konkrétnu verziu snortu (tar_path). Potom som vytvoril príkaz v Crontabe, aby sa spúšťala aktualizácia pravidiel presne v čase: 22:55

```
55 22 * * * pulledpork.pl -c pulledpork.conf -i disablesid.conf -T -H
```

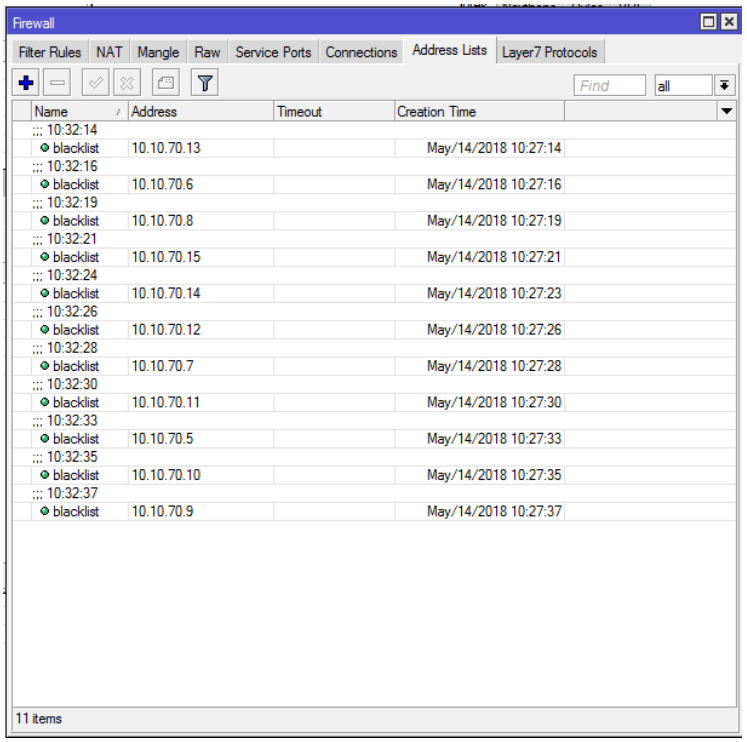
2.8 Test systému s pravidlami od pulledporku

Po vymazaní mnou vytvorených pravidiel som sa pustil k testovaniu systému s pravidlami stiahnutými zo stránok snortu za pomoci programu PulledPork. Test som začal zapojením mikrotiku, linuxového serveru a generátora útokov(SPIRENT Avalanche 3100) do jednoduchého zapojenia. Následne som spustil Packet Sniffer a program snort, otvoril v druhom terminály súbor syslogu a na druhom PC sledoval cez

winbox konfiguráciu mikrotiku, jeho logy a prácu s blokovanými IP adresami. Prvý test bol na blokovanie ICMP paketov, ako pri práci s vlastnými pravidlami. Test som zahájil s útokmi od 11 IP adries bez legitímnej komunikácie. Hneď po zahájení útoku som mohol pozorovať v súbore syslog, ako program snort detekuje prichádzajúcu komunikáciu a definuje ju ako ICMP pakety. Po 30 sekundách som mohol pozorovať, ako sa mi pomaly pridávajú všetky IP adresy do Blacklistu na zariadení mikrotik.



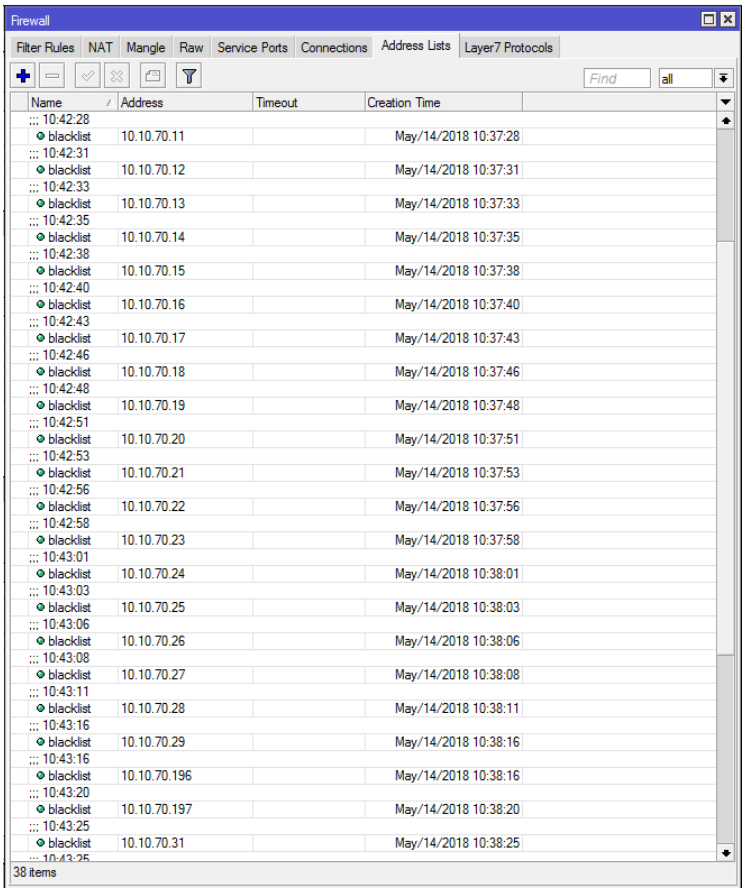
Obr. 2.23: Do syslogu sa zo snortu zapisuje hlásenia o útokoch ICPM



Obr. 2.24: Mikrotik blokuje 11 IP adries

Pri kontrole výstupov z mikrotiku obr.2.24 je možné hneď odpozorovať, že zapisovanie do blacklistu trvá 23 sekúnd. Toto zistenie bolo pre riešenie problému s obmedzovaním prístupu pre útočníka nepostačujúce. Preto bolo potrebné tento test

zopakovať. Teslotval som znova rovnaký druh útoku, bez legitímnej komunikácie, avšak som zväčšil počet útočníkov.



Obr. 2.25: Mikrotik blokuje 250 IP adries

Pri tomto teste s 250 útočníkmi je možné vidieť, že pre zápis do „adress listu“sa zapisovali IP adresy útočníkov veľmi pomaly.Čas medzi zapísaním prvej a poslednej IP adresy bol viac ako 10 minút. Toto bolo spôsobené nesprávnym vytvorením PHP skriptu. PHP skript sa spúšťa každých 30 sekúnd a prehľadáva sa ním syslog za posledných 30 sekúnd. Nachádza IP adresy, ktoré potom postupne predáva mikrotiku cez SSH spojenie. Avšak, toto spojenie naväzuje a ukončuje pre každú jednu IP adresu, ktorú predáva mikrotiku. Na základe toho bolo treba spraviť úpravu skriptu. Skript bolo treba upraviť tak, aby sa SSH spojenie vytváralo menej často, pričom by sa do mikrotiku zapísalo viac IP adries.

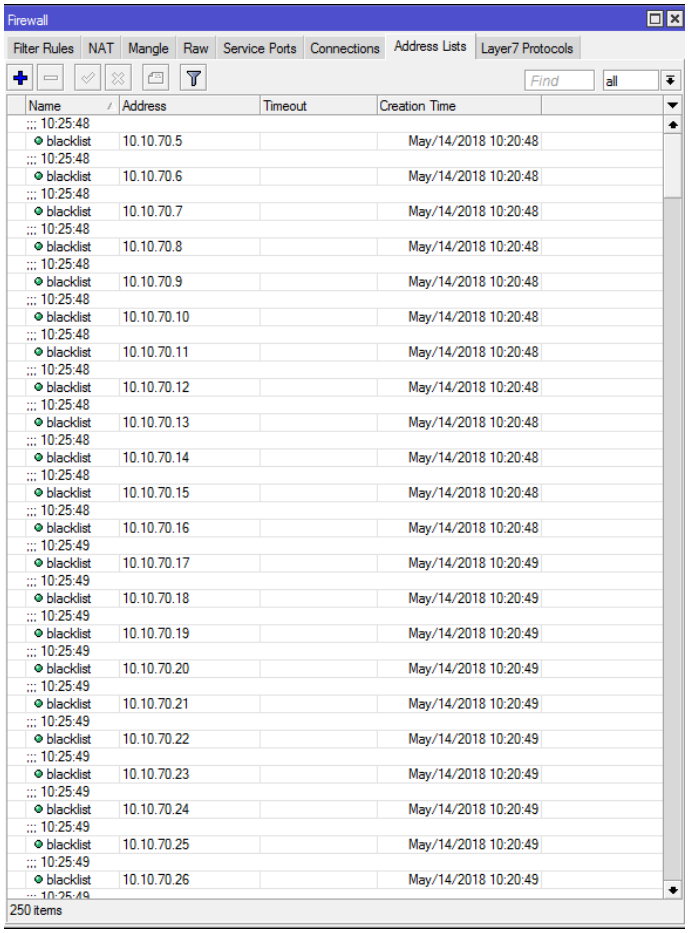
2.8.1 Úprava skriptu

Úpravu skriptu som vytváral teda preto, aby sa pri pri pridávaní IP adresy cez SSH komunikáciu nevytvára veľmi dlhá odozva. Vytvoril som premennú „$toBlock“ktorú som potom použil na zapisovanie len IP adries, ktoré treba prenášať na mikrotik.

Ďalšou potrebnou bola úprava funkcie „sendMikrotik“, kde bolo treba zmeniť prihlasovanie a odhlasovanie ssh komunikácie po zadaní iba jednej IP adresy. Takže som to zadal ako naviazanie komunikácie, na ktorú sa potom z poľa „$toBlock“postupne povyťahovali IP adresy a popridávali do „adress listu “mikrotiku. A.8

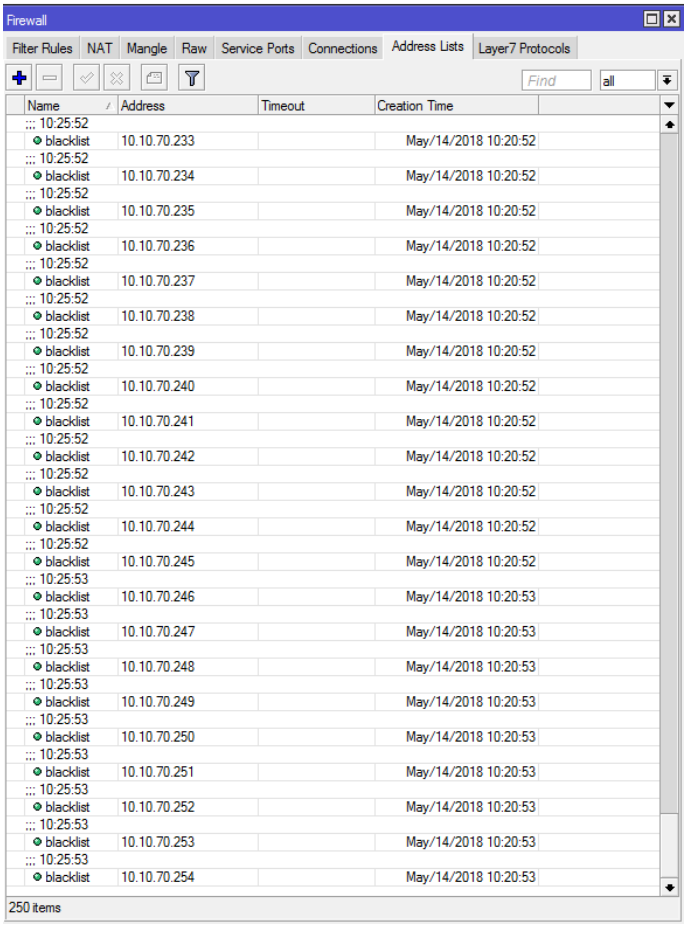
2.8.2 Výsledky meraní po úprave skriptu

Po úprave PHP skriptu som opäť zapojil systém a spustil všetky programy a doplnky pre správny chod. Spustil som generátor útokov SPIRENT Avalanche 3100, aby generoval ICMP FLOOD útok tj. ping útok, ktorého úlohou je zahltenie cieľového zariadenia ICMP paketmi. Útok bol generovaný na čas 3 minúty(180 sekúnd) z 250 IP adries na linuxový server, kde je spustený program snort. Zaznamenávanie IP adries blokovaných na mikrotiku som kontroloval cez winbox.



Obr. 2.26: Mikrotik začína blokovať IP adresy

Po zmene PHP skriptu, odozva v zapisovaní systému cez SSH spojenie do zariadenia Mikrotik trvala rádovo sekundy pre všetky zápisy. To znamená, že pri jednorázovom vytiahnutí záznamov zo syslogu za pomoci upraveného PHP skriptu a



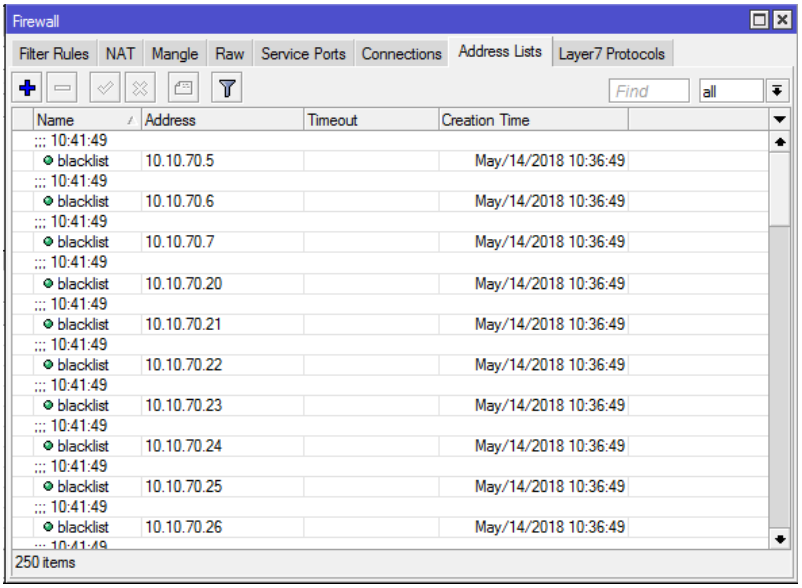
Obr. 2.27: Mikrotik blokuje 250 IP adries do pár sekúnd

následnom otvorení spojenia na mikrotik, zapíše systém a obmedzí prístup všetkým IP adresám(250tim) za 4 sekundy. Po vytvorení záznamov do mikrotiku (pridanie všetkých IP adries vyhodnotených PHP skriptom ako útoky), konkrétne do „adress listu“sa SSH pripojenie ukončí a čaká sa na ďalšie vytvorenie spojenia.

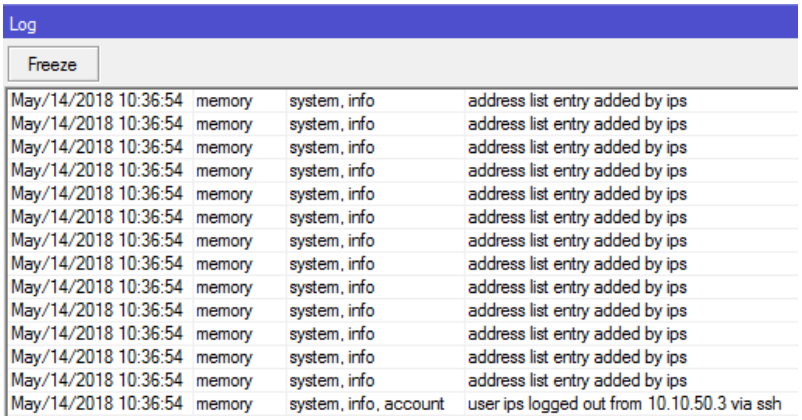
2.8.3 Test s legitímnou komunikáciou

Vytvoril som ďalší test s útokom generovaným na čas 3 minúty(180 sekúnd) z 250 IP adries na linuxový server, pričom som pridal vytváranie legitímnej komunikácie za pomoci posielania požiadavky na Apache2 server (HTTP_GET) na 10.10.50.3/index.html. Z adries 10.10.70.2-4 sa generujú požiadavky na stiahnutie default stránky apache2 servera každú sekundu. Z adries 10.10.70.5-254 sa generujú ICMP FLOOD útoky 4 každú sekundu. Z toho vyplýva, že sa Vykonáva 1004 požiadaviek v jednej sekunde. Pre spustenie útokov na zariadení SPIRENT Avalanche 3100 je nastavených prvých 30 sekúnd pre nadviazanie komunikácie a stanovenie siete za pomoci STP protokolu, ďalších 180 sekúnd pre útoky s legitímnou komunikáciou a posled-

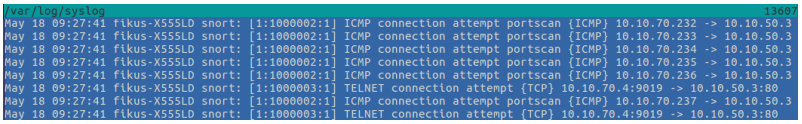
ných 10 sekúnd na odpájanie a ukončovanie útokov. V následujúcich obrázkoch je možno vidieť, ako sa IP adresy ICMP FLOODu blokujú a je taktiež viditeľné, že sa posiela aj legitímna komunikácia, avšak sa nesprávne detekuje ako TELNET a nie HTTP komunikácia.



Obr. 2.28: Blokovanie 250 adries generujúcich ICMP FLOOD



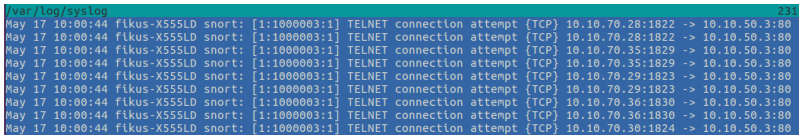
Obr. 2.29: Pridávanie IP adries do na mikrotiku(log z mikrotiku)



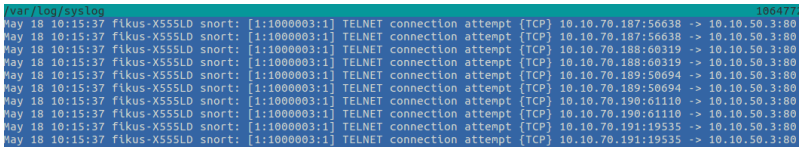
Obr. 2.30: Syslog a logovanie ICMP paketov a paketov HTTP

2.8.4 Testy iných útokov

Pre iné útoky ako sú **DDoS HTTP FLOOD** a **DDoS SYN FLOOD** systém zaznamená, avšak vyhodnotí kvôli sade pravidiel nesprávne a teda ako TELNET.

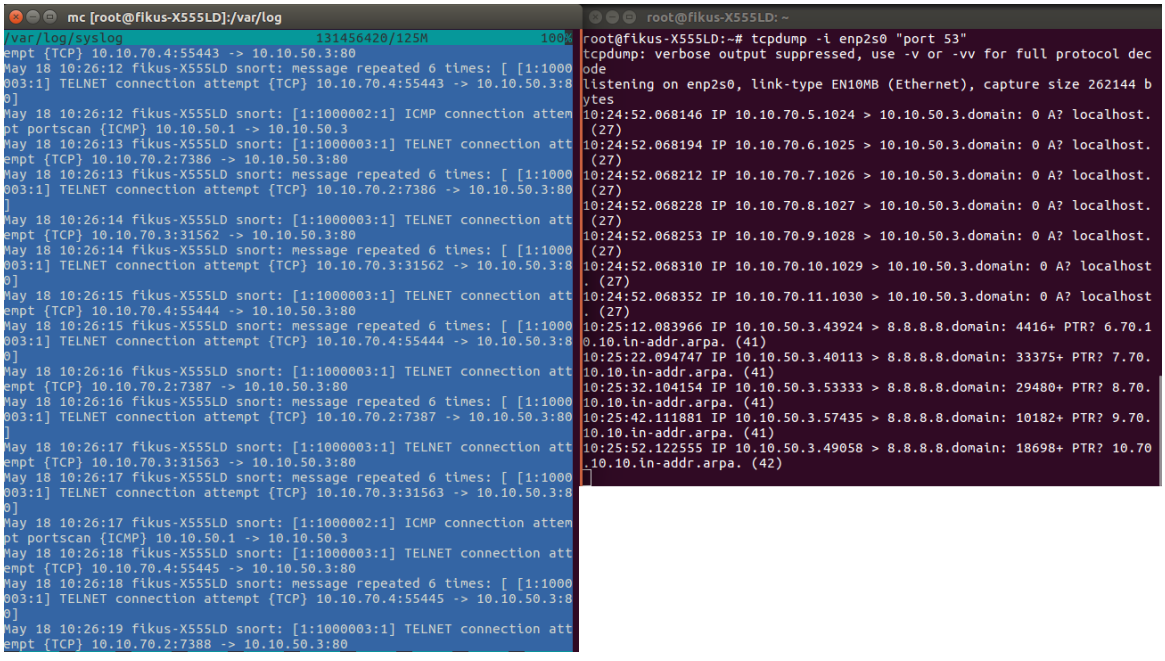


Obr. 2.31: Syslog zobrazuje snort záznam HTTP FLOODu ako TELNET



Obr. 2.32: Syslog zobrazuje snort záznam SYN FLOODu ako TELNET

Pre útok **DNS FLOOD** na port 53 sa za pomoci programu snort nepodarilo prísť. Pri prezeraní komunikácie za pomoci „tcpdump“je možno prichádzajúce pakety vidieť, avšak výstup zo snortu nehlási žiadnu podozrivú aktivitu.



Obr. 2.33: Syslog nezobrazuje žiadne snort záznamy DNS FLOODu

2.8.5 Zmena pravidiel

Po neúspešnom detekovaní viacerých útokov som sa rozhodol zmeniť pravidlá. Pravidlá automatický sťahované som vymenil za pravidlá jednorázovo stiahnuté zo stránok snortu za pomoci odkazu (číslo na konci je Oinkcode, ktorý je originál pre každého registrovaného užívateľa):

`https://www.snort.org/rules/snortrules-snapshot-29111.tar.gz?oinkcode=c209acaac1573ee.....`

Obr. 2.34: Stiahnutie pravidiel pre snort

V súbore ktorý som stiahol boli tri zložky s názvami:

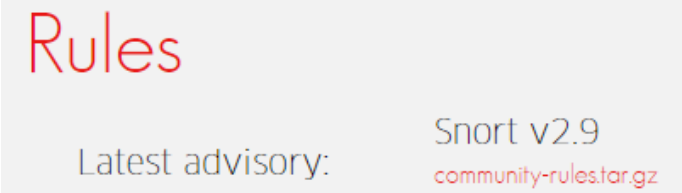
- preprocesor_rules
- rules
- so_rules.

V štruktúre adresárov snortu som potom tieto tri zložky nakopíroval do /etc/snort/rules/rules, aby som sa neplietli s pravidlami automaticky sťahovanými. Následne som pozmenil snort.conf a to tak, že som $RULE_PATH rozšíril z /etc/snort/rules na /etc/snort/rules/rules, aby mal snort správne zadefinovanú cestu k novým pravidlám. Potom som musel odkomentovať zahrňovanie pravidiel v snorte (zmazaním znaku „#“pred príkazom „include“). Nutnosť bola vypnúť (zakomentovať) staré, automaticky sťahované pravidlá v snort.conf súbore (include $RULE_PATH/snort.rules).

Test so statickými pravidlami od snortu sa mi ale útoky takisto nepodarilo detekovať. Preto som sa rozhodol vyskúšať variantu komunitných pravidiel, ktoré sú zdarma a voľne dostupné. Na stránkach snortu som teda stiahol „community rules“.

Odkaz na priame stiahnutie,a lebo možno vidieť na obrázku odkaz na stránke snortu.org/downloads:

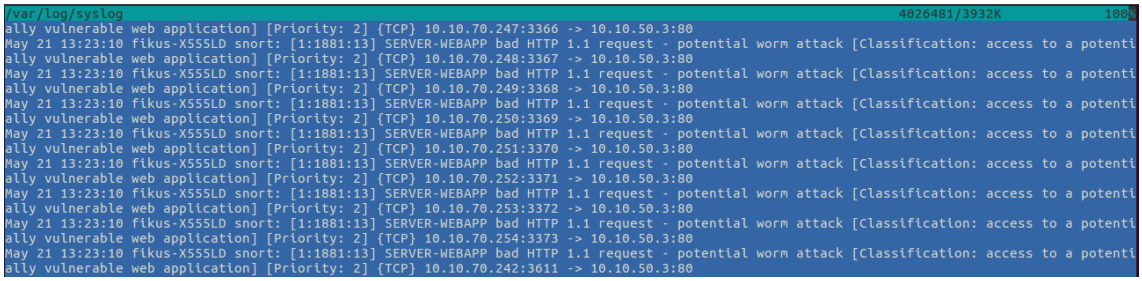
`https://www.snort.org/rules/community`



Obr. 2.35: Odkaz na stiahnutie komunitných pravidiel

2.8.6 Test so statickými pravidlami

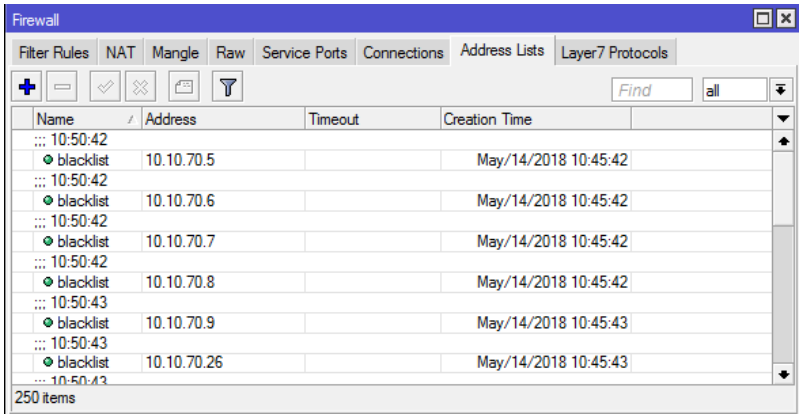
Pri tomto teste som znova zapojil všetky zariadenia a spustil programy a súčasti potrebné pre testovanie. Začal som testom ICMP FLOODu, ktorý rovnako ako pri pravidlách automaticky sťahovaných pomocou pulledporku snort vedel odchytávať a po 30 sekundách(spustení PHP skriptu) zapisovať a blokovať pomocou uloženia IP adries do blacklistu na mikrotiku.
Následne som sa venoval testovaniu HTTP FLOODu, ktorý posiela požiadavky GET, s tým že hlavička nie je úplná a teda sa iba snaží zahltiť server. Tieto GET požiadavky sú posielané znova na apache2 server na adresu 10.10.50.3 a port 80. Ide o nadväzovanie 1000 fiktívnych užívateľov z 250 IP adries a legitímnej komunikácie s reálnou požiadavkou GET s úplnou hlavičkou na sťahovanie stránky defaul toho istého apache2 servera. Logovací súbor apache2 servera z testu a nadväzovania spojenia požiadavkou GET je priložený na CD.



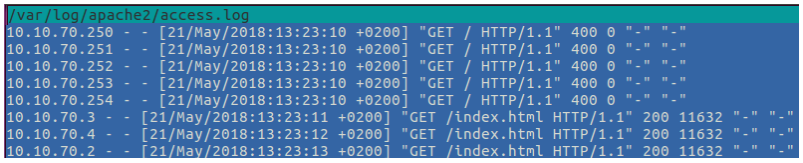
Obr. 2.36: Syslog výpis HTTP FLOODu

Detekcia programom snort s komunitnými pravidlami odhalila, že ide o chybnú HTTP 1.1 požiadavku a následne ju blokovala, odstup od začiatku simulovaného

útoku od blokovania je asi 32 sekúnd. Tento čas sa následne dá ešte znížiť zmenšením času za ktorý sa prehľadáva syslog a zvýšením počtov spúšťania skriptu „php.php“. Pre tento prípad však čas 32 sekúnd je braný ako dostačujúci. Pri výpise z logu

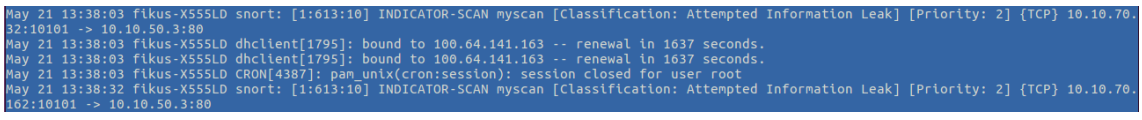


Obr. 2.37: Blokovanie 250 IP adries pri HTTP FLOODe



Obr. 2.38: Logovaci súbor zo servera apache2

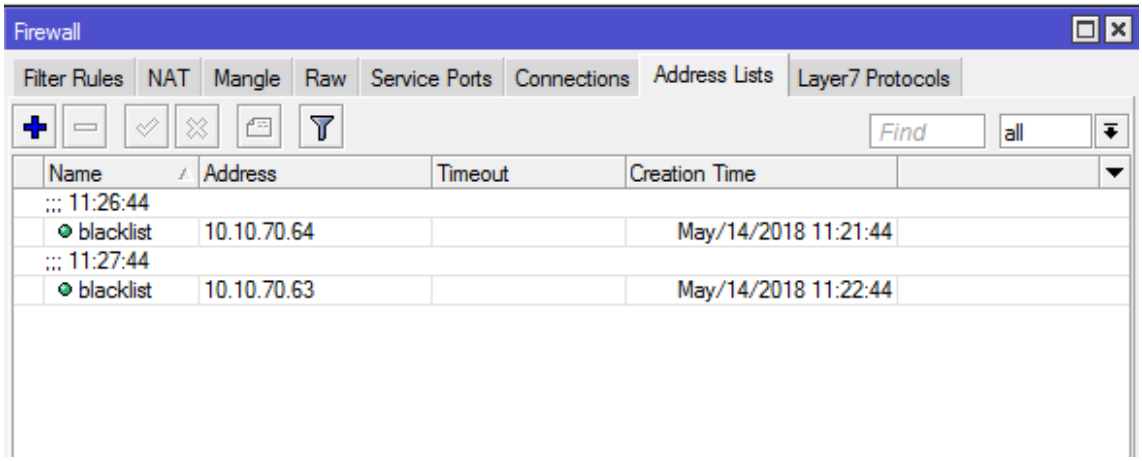
servera apache2 je možné vidieť, do času kedy sú IP adresy útočníka blokované, že reálne požiadavky prídu až na server. Adresy 10.10.70.2-4 (legitímna komunikácia) sú taktiež viditeľné, ale ako reálne požiadavky na stiahnutie stránky.
Útok s názvom SYN FLOOD sa podarilo zaznamenať iba čiastočne, pričom sme odchytili iba 4 IP adresy a tieto neboli vyhodnotené ako SYN FLOOD, ale ako SCAN. K lepšiemu zachytávaniu tohoto druhu útoku by bolo nutné vytvoriť podmienky v pravidlách. Pri tomto druhu útokov ide o nadväzovanie spojenia a teda by bolo potrebné zaznamenávať počet pokusov o nadviazanie spojenia.



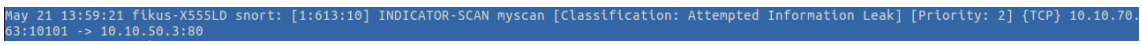
Obr. 2.39: Snort s pravidlami zle vyhodnotil SYN FLOOD útoky ako SCAN

Posledným testovaným útokom bol RST FLOOD. Počas útoku RST FLOOD je server bombardovaný falošnými paketmi RST, ktoré nemajú žiadnu spojitosť so

žiadnou z relácií uložených v databáze servera. Server musí zničiť veľa systémových prostriedkov aby sa s tým vysporiadal a tak sa zaťažuje najmä pamäť RAM a procesor. Tento druh útoku sa nám podobne ako SYN FLOOD s komunitnými pravidlami nepodarilo odstrániť a tak sme čiastočne začali zaťažovať procesor. Snort znovu zaradil dve z IP adries počas tohoto testu do blacklistu, boli znova posúdené nesprávne ako SCAN. Počas týchto testov však všetká legitímna komunikácia prebiehala bez problémov.



Obr. 2.40: Snort s pravidlami zle vyhodnotil RST FLOOD útoky ako SCAN



Obr. 2.41: Snort s pravidlami zle vyhodnotil RST FLOOD útoky ako SCAN

3 ZÁVER

Cieľmi tejto semestrálnej práce bolo zoznámiť sa s problematikou zabezpečenia sietí s prvkami Mikrotik a servermi s OS linux. Na základe prvotnej analýzy navrhnúť a následne aj realizovať zabezpečenie sietí s prvkami pre detekciu a analýzu útokov. V prvej kapitole som sa zoznamoval s problematikou zabezpečenia sietí. Prechádzal som firewalmi, ich delením, až k firewallom s integrovanými súčasťami IDS/IPS. Následne som zisťoval možnosti ako a kde používať IDS a IPS systémy. Vybral som si dve najznámejšie varianty Snort a Suricatu pre bližšie skúmanie. Kvôli dostupnejším informáciám, náväznosti na ďalšie systémy ako je snortby(grafické prostredie) alebo rozšírené logovanie cez databázy a spravovanie, jednoduchšiemu prevedeniu, ako aj faktu, že pravidlá pre program snort sú lepšie spracované a dostupnejšie v otestovaných verziách, som sa rozhodol pre realizáciu systému s programom snort. Následne som začal s realizáciou. V prvom kroku som si zohnal potrebné vybavenie pre hardwarovú realizáciu, pretože v dalšej časti som chcel otestovať reálne možnosti a schopnosti tohoto systému. Následne som začal s upgrade-om RouterOS a následným zapracovaním na topológii. Nastavil som Mikrotik na základnú konfiguráciu, aby som mohol pokračovať. Potom bolo teba nastaviť Paket Sniffer na konkrétny port, aby vysielal všetku komunikáciu, ktorá mu príde z WAN siete 2.3. Pokračoval som s výberom distribúciu Linuxu a jej inštaláciou. Zisťoval a nahrával potrebné balíčky pre Ubuntu 16.04.3 x64(otestoval som aj systém Ubuntu 18.04). Akonáhle som mal základnú konfiguráciu Linuxu, začal som riešiť program Snort, jeho inštaláciu, podporu a konfiguráciu. Po úspešnej inštalácii Snortu a balíčkov spolu s presúvaním a vytváraním priečinkov a súborov som sa mohol venovať pravidlám. V Konfiguračnom súbore snortu v adresáry /etc/snort/snort.conf som si musel odkomentovať implementáciu súbor etc/snort/rules/local.rules kde som následne mohol zapísať pravidlo na testovanie. Po dlhšom ladení som zistil ako príkazy zapísať a spúšťať Snort spolu s nástrojom, ktorý zakódované dáta z Mikrotiku dekóduje. Po tomto všetko som sa dostal k vypisovaniu chýb do súboru /var/log/snort/alert A.3. Cieľ logovania a zápisu z programu snort bol ale syslog. Následne som teda zmenil konfiguráciu konfiguračného súbora snortu a vytvoril záznam v novovytvorenom súbore auth.conf, kvôli smerovaniu na syslog. Ďalším krokom bolo vytvorenie spojenia, aby sa záznamy zo syslogu dostávali na mikrotik. Bolo treba vytvoriť niekoľko skriptov, aby spolupracovali, otvárali spojenie SSH na mikrotiku a ukladali IP adresy útočníkov do blacklistu. Na mikrotiku bol vytvorený systém, ktorý na jednej strane zapisoval IP adresy do blacklistu a na strane druhej ich z blacklistu vymazával. Toto má zabrániť pri zle vyhodnotenom útoku, aby sa mohla znova nadviazať komunikácia. Po otestovaní systému nastali dva vážnejšie zdržania, kde som musel vytvárať úpravu PHP skriptu, ktorý zapisoval IP adresy veľmi pomaly a sada pravidiel, ktorá

pre naše testované útoky nefungovala. Zapríčinené to bolo otváraním SSH spojenia pre každý zápis do blacklistu. Tento problém som opravil zmenou skriptu, aby SSH spojenie nadväzovalo až po načítaní IP adries do tzv. buffera, ktorý som spravil z obyčajnej premennej typu pole. Druhý problém som riešil stiahnutím inej sady pravidiel a keď nepomáhali ani tie, využil som možnosť použitia voľne dostupnej sady pravidiel od komunity.

Systém som otestoval pre typy útokov: ICMP FLOOD, HTTP FLOOD, SYN FLOOD a RST FLOOD. Testoval som zabraňovanie útokom, pričom sa mi generoval aj legitímna komunikácia vo forme požiadavky GET na apache2 server na prebratie stránky „defaul“. ICMP FLOOD a HTTP FLOOD sa bez väčších problémov nakoniec podarilo odbúravať v čase približne 32 sekúnd od začiatku útoku. Tento čas je možné ešte znížiť zmenou času prehľadávania syslog súboru skriptom PHP. Optimalizovať čas, ktorý je možno nastaviť som z časových príčin netestoval. Pre útoky typu SYN FLOOD a RST FLOOD sa mi nepodarilo zhotoviť pravidlo a ani odchytávanie, či zaznamenávanie komunikácie vyhodnotenej ako útok.

LITERATÚRA

[1] *Cisco*What is a firewall? [online]. Dostupné z URL: <http://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html>.

[2] *Pina Chhatrala* Types of firewall, 28.September (2015) [online]. Dostupné z URL: <https://www.slideshare.net/PinaChhatrala1/types-of-firewall-53269158>

[3] Ing. Jan Jeřábek, Ph.D.(2015) Pokročilé komunikační techniky str.74 *Brno* Dostupné z URL, po prihlásení: <https://www.vutbr.cz/www_base/priloha.php?dpid=67088>

[4] *T. Paulus* „Spustite sieť! 3“– Systémy detekcie a prevencie prieniku 29.Jul(2004) [online]. Dostupné z URL: <http://preventista.sk/info/spustite-siet-3-systemy-detekcie-a-prevencie-prieniku/>.

[5] *KAZIENKO, Przemyslaw a Piotr DOROSZ.* Intrusion Detection Systems (IDS) Part I – (network intrusions; attack symptoms; IDS tasks; and IDS architecture) [online]. TechGenix Limited, 1997, 2003 [cit. 2015-12-09]. Dostupné z URL: <http://techgenix.com/Intrusion_Detection_Systems_IDS_Part_I__network_intrusions_attack_symptoms_IDS_tasks_and_IDS_architecture/>.

[6] *Cyberpedia* , What is an intrusion prevention system [online]. Dostupné z URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips>.

[7] *Cyberpedia* , What is an intrusion detection system [online]. Dostupné z URL: <https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids>.

[8] Informatika pre samoštúdium [online]. Dostupné z URL: <http://www.tonko.eu/ite/Web-bezpecnost/Detekcia>.

[9] *Tony Bradley* , Introduction to Intrusion Detection Systems (IDS) 27. August (2017)[online]. Dostupné z URL: <https://www.lifewire.com/introduction-to-intrusion-detection-systems-ids-2486799>.

[10] *Steve Schupp* Ron Lepofsky, 1. December (2000) [online]. Dostupné z URL: <https://www.networkworld.com/article/2228598/security/intrusion-detection--why-do-i-need-ids--ips--or-hids-.html>.

[11] Intrusion detection system[online]. Dostupné z URL: <https://www.giac.org/paper/gsec/235/limitations-network-intrusion-detection/100739>.

[12] John R. Vacca (2010). Managing Information Security. Syngress. page 137. ISBN 978-1-59749-533-2. Retrieved 29 June 2010. *Kniha: Recent Advances in Intrusion Detection* [online]. Dostupné z URL: <https://books.google.sk/books?id=uwKkb-kpmksC&pg=PA137&redir_esc=y#v=onepage&q&f=false>.

[13] *Tomáš Paulus*, Prehlad IPS/IDS (2014)[online]. Dostupné z URL: <http://preventista.sk/info/spustite-siet-3-systemy-detekcie-a-prevencie-prieniku/>.

[14] *Tony Bradley*Lifewire, Ids and Ips software, 7.July (2017)[online]. Dostupné z URL: <https://www.lifewire.com/ids-and-prevention-ips-software-2487316>

[15] The Open Information Security Foundation.[online]. Dostupné z URL: <https://suricata-ids.org/>

[16] SAMMONS, John. *The basics of digital forensics: the primer for getting started in digital forensics.* 2nd edition. Waltham, MA: Elsevier, 2015. 177 s. ISBN 978-0-12-801635-0.

[17] *Social engineering*[online]. Dostupné z URL: <https://en.wikipedia.org/wiki/Social_engineering_(security)>

[18] Wiki.Mikrotik, 15. February (2014) <https://wiki.mikrotik.com/wiki/Mikrotik_IPS_IDS>

[19] *de.Wikipedia* <https://de.wikipedia.org/wiki/Snort>

[20] *Doug Dineley*, The greatest open source software of all time, 17.August (2009) <https://www.infoworld.com/article/2631146/open-source-software/the-greatest-open-source-software-of-all-time.html>

[21] *Noah Dietrich*, Snort 2.9.9.x on Ubuntu – Part 1: Installing Snort <http://sublimerobots.com/2017/01/snort-2-9-9-x-ubuntu-installing-snort/>

[22] *Wikipedia*, the free encyclopedia October (2007) <https://en.wikipedia.org/wiki/TZSP>

[23] *Admins's Choice*, Magazine for System Admins & Technologists (2013) <http://www.adminschoice.com/crontab-quick-reference>

[24] *mc0e* Serverfault fórum (2014) <https://serverfault.com/questions/636770/snort-not-sending-alert-log-file-to-syslog-server?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa>

[25] *Robert Penz*Howto fix -bash: ./trafr(2014) <https://robert.penz.name/index.php?s=trafr>

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

IDS Intrusion detection system

IPS Intrusion prevention system

NIDS Network Intrusion Detection Systems

TCP Transmission Control Protocol

UDP User Datagram Protocol

ICMP Internet Control Message Protocol

HIDS Host-based Intrusion Detection Systems

DoS Denial-of-Service

DDoS Distributed Denial of Service

DHCP Dynamic Host Configuration Protocol

NAT Network Address Translation

WAN Wide Area Network

LAN Local Area Network

TZSP TaZmen Sniffer Protokol

ZOZNAM PRÍLOH

A ZOZNAM PRÍLOH

A.1 Vytvorenie priečinkov a súborov

Vytvorí priečinky: sudo mkdir /etc/snort
sudo mkdir /etc/snort/rules
sudo mkdir /etc/snort/rules/iplists
sudo mkdir /etc/snort/preproc_rules
sudo mkdir /usr/local/lib/snort_dynamicrules
sudo mkdir /etc/snort/so_rules

Vytvorí priečinky, kde suú uložené pravidlá a IP zoznamy
touch /etc/snort/rules/iplists/black_list.rules
touch /etc/snort/rules/iplists/white_list.rules
touch /etc/snort/rules/local.rules
touch /etc/snort/sid-msg.map

Vytvorí priečinky pre logy:
mkdir /var/log/snort
mkdir /var/log/snort/archived_logs

Priradenie oprávnení:
chmod -R 5775 /etc/snort
chmod -R 5775 /var/log/snort
chmod -R 5775 /var/log/snort/archived_logs
chmod -R 5775 /etc/snort/so_rules
chmod -R 5775 /usr/local/lib/snort_dynamicrules

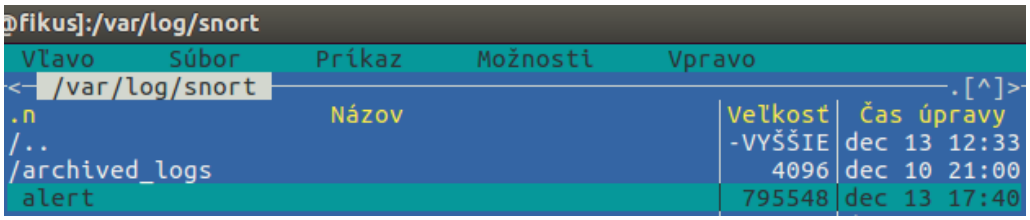
Zmena vlastníka:
chown -R snort:snort /etc/snort
chown -R snort:snort /var/log/snort
chown -R snort:snort /usr/local/lib/snort_dynamicrules

A.2 Prekopíroavnie súborov

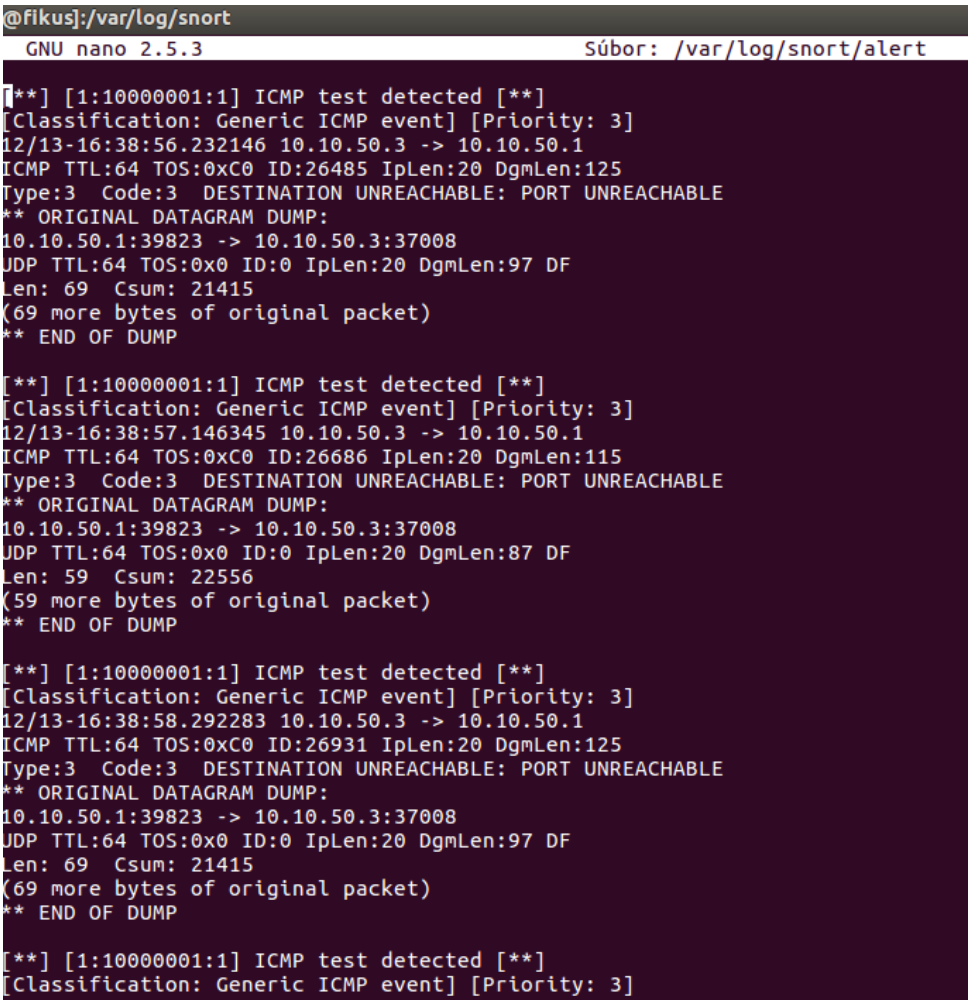
cd /snort_src/snort-2.9.11/etc/
cp *.conf* /etc/snort
cp *.map /etc/snort
cp *.dtd /etc/snort

cd /snort_src/snort-2.9.11/src/dynamic-preprocessors/build/usr/local/lib/snort_dynamicpreprocessor/
cp * /usr/local/lib/snort_dynamicpreprocessor/

A.3 Alerty



Obr. A.1: Aletry v Midnight Commandery



Obr. A.2: Aletry v programe nano

A.4 Skript na strane IPS

```
<?php
$blocked=array();
exec('cat /var/log/syslog | grep "`date -d "-30 second"
 "+\%e \%H:\%M"`"',$lastMin);

foreach($lastMin as $line) {
   if (strpos($line,"Priority:")!==FALSE || strpos($line,"portscan")!==FALSE)
preg_match("/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/", $line, $matches);
   $filter=$matches[0];
   if (!in_array($filter, $blocked))
   {
   $blocked[]=$filter;
if (strpos($filter,"10.10.50.")!==FALSE) continue;
sendMikrotik('10.10.50.1', 'ips', 'snort',$filter);
   }
   }

}
function sendMikrotik($mt,$user,$pass,$filter) {
$connection = ssh2_connect($mt);
ssh2_auth_password($connection,$user,$pass);
sleep(1);
$stream = ssh2_exec($connection, ':global ip '.$filter);
$stream = ssh2_exec($connection, '/system script run filter');
$stream = ssh2_exec($connection, 'quit');
}
```

A.5 Mikrotik skript

```
:global ip

local time ([/system clock get time]+("00:05:00"))
if ($time > "23:59:59") do={
  :local time "00:05:00"
}
/ip firewall address-list add list=blacklist address=$ip comment=$time
```

A.6 Mikrotik skript

```
:local currentTime [/system clock get time]

foreach i in=[/ip firewall address-list find list=blacklist] do={
:local comment [/ip firewall address-list get $i comment]
:local ip [/ip firewall address-list get $i address]
  :if ( $comment < $currentTime ) do={
      /ip firewall address-list remove [find address=$ip]
   }
}
```

A.7 PulledPork

```
Riadok 19:  odkomentovať a namiesto <oinkcode> vložiť kontrétny kód a
<> nepoužiť
Riadok 74:  zmeniť na: rule_path=/etc/snort/rules/snort.rules
Riadok 89:  zmeniť na: local_rules=/etc/snort/rules/local.rules
Riadok 92:  zmeniť na: sid_msg=/etc/snort/sid-msg.map
Riadok 96:  zmeniť na: sid_msg_version=2
Riadok 119:  zmeniť na: config_path=/etc/snort/snort.conf
Riadok 133:  zmeniť na: distro=Ubuntu-12-04
Riadok 141:  zmeniť na: black_list=/etc/snort/rules/iplists/black_list.rules
Riadok 150:  zmeniť na: IPRVersion=/etc/snort/rules/iplists
```

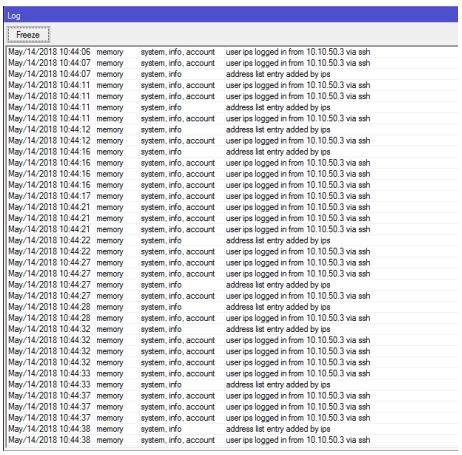
A.8 Skript po uprave

```
<?php
$blocked= array();
$toBlock = array();
exec('cat /var/log/syslog | grep "`date -d "-30 second"
"+\%e \%H:\%M"`"',$lastMin);

foreach($lastMin as $line) {
   if (strpos($line,"Priority:")!==FALSE ||
strpos($line,"portscan")!==FALSE)
   {
preg_match("/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/", $line, $matches);
   $filter=$matches[0];
```

```
   if (!in_array($filter, $blocked))
   {
   $blocked[]=$filter;// vsetky blokovane aj 50.1 ...
   if(strpos($filter,"10.10.50.")===FALSE){
$toBlock[]= $filter;
}
}
}
}
sendMikrotik('10.10.50.1', 'ips', 'snort',$toBlock);
function sendMikrotik($mt,$user,$pass,$toBlock) {
$connection = ssh2_connect($mt);
ssh2_auth_password($connection,$user,$pass);
sleep(1);
foreach ($toBlock as $filter){
$stream = ssh2_exec($connection, ':global ip '.$filter);
$stream = ssh2_exec($connection, '/system script run filter');
}
$stream = ssh2_exec($connection, 'quit');
}
```

A.9 Otvaranie SSH pripojenia na mikrotik



Obr. A.3: Otvárania SSH pripojenia na mikrotik

B OBSAH PRILOŽENÉHO DVD